

Operating Guide

AK-SM8xxA 3.3 and higher

Use Header Authentication

Description

To secure, that XML request are only valid and possible with username and password a new feature called **Use Header Authentication** has been added.

How to enable and use this feature

Under Configuration / Comm this new feature can be enabled

Use HTTPS	Yes	▼
Use Secure TLS	Yes	▼
Use Header Authentication	Yes	▼

After a change, the SM800A must be initialized.

Press to Initialize

It's a base64 encoded user and password in the header.

How to use it:

Example made with the XML tool postman.

On any header an authentication key called – **aksm-auth** – is needed, as soon the feature is enabled.

Key	Value	Description
<input checked="" type="checkbox"/> Postman-Token	<calculated when requ...	
<input checked="" type="checkbox"/> Content-Type	application/xml	
<input checked="" type="checkbox"/> Content-Length	<calculated when requ...	
<input checked="" type="checkbox"/> Host	<calculated when requ...	
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.32.2	
<input checked="" type="checkbox"/> Accept	/*/*	
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection	keep-alive	
<input checked="" type="checkbox"/> aksm-auth	Basic ZGFuOmFBQDEy...	

The value can be created here:

<https://www.blitter.se/utills/basic-authentication-header-generator/>

and also here:

<https://www.debugbear.com/basic-auth-header-generator>

Without doing it correct or the header is missing, the response will look like this:

Content-Length: 12
Unauthorized

Changes on AK-SM 8xxA version 4 and higher

Due to security, there are 3 new settings in the front end in a new menu.

You can find the menu below Configuration/Security

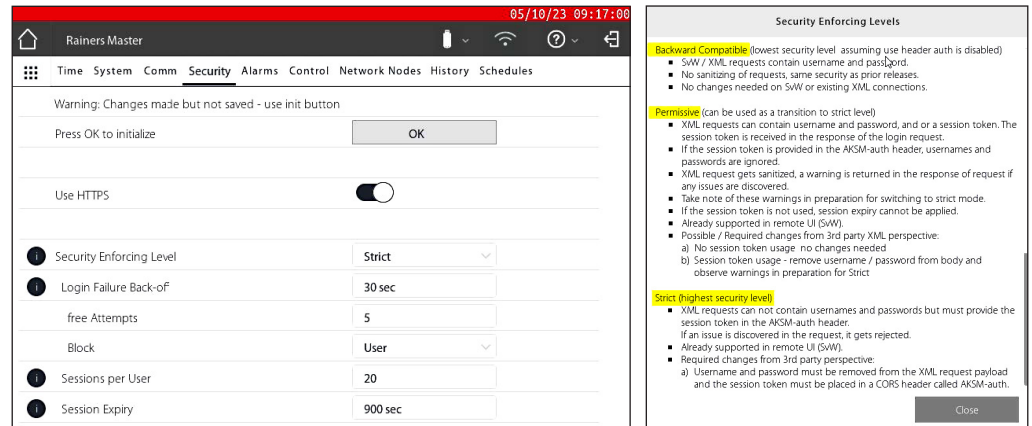
From now on it's possible to change the level of security.

Level 1 = Backwards Compatible

Level 2 = Permissive

Level 3 = Strict

The explanation can be seen on local screen, behind the info  button.



The screenshot shows the 'Security Enforcing Levels' configuration page in the Rainers Master interface. The page includes a warning message, a table of settings, and a detailed explanation of the security levels.

Setting	Value
Security Enforcing Level	Strict
Login Failure Back-off	30 sec
free Attempts	5
Block	User
Sessions per User	20
Session Expiry	900 sec

Security Enforcing Levels

Backward Compatible (lowest security level - assuming use header auth is disabled)

- SW / XML requests contain username and password.
- No sanitizing of requests, same security as prior releases.
- No changes needed on SW or existing XML connections.

Permissive (can be used as a transition to strict level)

- XML requests can contain username and password, and/or a session token. The session token is received in the response of the login request.
- If the session token is provided in the AKSM-auth header, usernames and passwords are ignored.
- XML request gets sanitized, a warning is returned in the response of request if any issues are discovered.
- Take note of these warnings in preparation for switching to strict mode.
- If the session token is not used, session expiry cannot be applied.
- Already supported in remote UI (SW).
- Possible / Required changes from 3rd party XML perspective:
 - No session token usage - no changes needed
 - Session token usage - remove username / password from body and observe warnings in preparation for Strict

Strict (highest security level)

- XML requests can not contain usernames and passwords but must provide the session token in the AKSM-auth header.
- If an issue is discovered in the request, it gets rejected.
- Already supported in remote UI (SW).
- Required changes from 3rd party perspective:
 - Username and password must be removed from the XML request payload and the session token must be placed in a CORS header called AKSM-auth.

No changes on the XML description is needed, if Level 1 or 2 is chosen.

Level 3 (Strict) needs a change.

The 1st request must be to authenticate with following command: ("x" and "y" must be replaced with the programmed credentials)

```
<cmd
  action="getauth"
  user="xxx"
  password="yyyy@yyyy" >
</cmd>
```

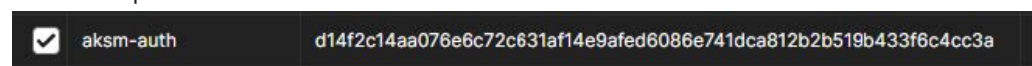
Following response with a new token will be received:

```
<?xml version="1.0" encoding="utf-8"?>
<resp action="getauth" error="0">
  <authorization>01-01</authorization>
  <language>English</language>
  <authtype>4194303</authtype>
  <access>01-01</access>
  <accesslevel>1</accesslevel>
  <strongpassword>1</strongpassword>
  <session_token>d14f2c14aa076e6c72c631af14e9afed6086e741dca812b2b519b433f6c4cc3a</session_token>
  <last_login>2023-10-05 08:08:48</last_login>
  <failed_attempts>0</failed_attempts>
</resp>
```

This new token must be used in the header of all coming requests and is guilty as long data are requested. On no activity the token is guilty for the chosen session expire time.

On any header an authentication key called – **aksm-auth** – is needed now with the new token.

In this example the header is like this:



The screenshot shows the HTTP header for the request. The key is 'aksm-auth' and the value is 'd14f2c14aa076e6c72c631af14e9afed6086e741dca812b2b519b433f6c4cc3a'.

If the session is outdated the reply looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<resp error="100"> No existing session for user/token.</resp>
```