

# APM: Variable Assign

## Variable assign best practice

### Username / Domain Management

get username

image-1615964871694.png

session.logon.last.username

extract CN from certificate subject and set it in username variable

```
set subject [split [mcget {session.ssl.cert.subject}] ",="];
foreach {name value} $subject {
  if {[string trim $name] equals "CN"} {
    return [string trim $value];
  }
}
```

session.logon.last.username

combine username and domain variables

```
expr { "[mcget {session.logon.last.domain}]\\"[mcget {session.logon.last.username}]" }
```

session.ad.last.attr.userPrincipalName

Get the UPN when using AD-Auth. This variable is filled out with the UPN from the AD and can be used in the VPE.

remove UPN from session.logon.last.username:

```
expr { [string range [mcget {session.logon.last.username}] 0 [expr [string first "@" [mcget {session.logon.last.username}] 0] -1] ] }
```

session.logon.last.ntdomain

extract NT domain name from logon name

```
if { [mcget {session.logon.last.username}] contains "\\" } {  
    set username [string tolower [mcget {session.logon.last.logonname}]];  
    return [string range $username 0 [expr {[string first "\\" $username] -1} ]];  
} else {  
    return {}  
}
```

one-line code

```
expr {[set username [string tolower [mcget {session.logon.last.logonname}]]] contains "\\" ? [string range  
$username 0 [expr {[string first "\\" $username] -1} ] ] : "" }
```

## session.logon.last.domain

static assignment from ntdomain

```
switch [string tolower [mcget {session.logon.last.ntdomain}]] {  
    "domain1" { return "domain1.local" }  
    "domain2" { return "domain2.local" }  
    default { return "default.local" }  
}
```

## session.logon.last.username

Extract username name from logonname (full username from logon page even if split domain from username is checked)

```
set username [string trim [mcget {session.logon.last.logonname}]];  
if { $username contains "\\" } {  
    return [string range $username [expr {[string first "\\" $username] +1} ] end ];  
} else { return $username }
```

## session.logon.last.upn

Extract UPN value from Certificate X509Extension

```
set extension [string tolower [mcget {session.ssl.cert.x509extension}]];  
return [string range $extension [expr {[string first "othername:upn<" $extension] +14} ] [expr {[string last ">"  
$extension] -1} ]];
```

# Session / Timeout Management

## session.inactivity\_timeout

Change inactivity session timeout based on a checkbox on the logon page (logon variable trusted)

```
if { [mcget {session.logon.last.trusted}] == 1 } { return {5400} } else { return {1800} }
```

one-line code (5400 seconds if condition before ? success, 1800 seconds else)

```
expr { [mcget {session.logon.last.trusted}] == 1 ? {5400} : {1800} }
```

## session.inactivity\_timeout

Change inactivity session timeout based on client type (iOS, Android and WindowsPhone : half of inactivity timeout configured in profile parameters)

```
expr { [mcget {session.client.platform}] == "WindowsPhone" || [mcget {session.client.platform}] == "Android" || [mcget {session.client.platform}] == "iOS" ? [mcget {session.inactivity_timeout}]/2 : [mcget {session.inactivity_timeout}] }
```

## session.max\_session\_timeout

force to close the session à 17:00

```
expr { [clock scan "17:00"] - [mcget {session.user.starttime}] }
```

## session.max\_session\_timeout

After a AD query which retrieve attribute logonHours, force to close the session when user at the end of allowed logon hours

```
set maximumSessionSeconds 604800
if {[set logonHours [mcget {session.ad.last.attr.logonHours}]] != "" && $logonHours != "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"} {
    #convert string to binary string
    binary scan [binary format H* $logonHours] b* logon_hours_binary_string
    # evaluate the number of seconds from last sunday
    set time_from_sunday [expr {[clock seconds] - [clock scan "last sunday"]}];
    # search in string next hours with 0 value
    set current_index [expr {$time_from_sunday / 3600}];
    # convert the index to number of seconds from last sunday
```

```
if {[set next_denied_index [string first 0 $logon_hours_binary_string$logon_hours_binary_string
$current_index]] == $current_index } {return 0}

# evaluate number on seconds to disconnect time

return [expr { $next_denied_index*3600 - $time_from_sunday}]

} else { return $maximumSessionSeconds}
```

## Windows Info

### session.windows\_info\_os.last.fqdn

search and return FQDN hostname in computer names list after **windows Info** Box

```
foreach x [split [mcget {session.windows_info_os.last.computer}] "|"] {
    if { $x ends_with ".f5demo.lab" } {
        return $x
    }
}
```

### session.windows\_info\_os.last.computer\_name

search FQDN hostname in computer names list after **windows Info** Box, then return shortname (without domain name)

```
foreach x [split [mcget {session.windows_info_os.last.computer}] "|"] {
    if { $x ends_with ".f5demo.lab" } {
        return [lindex [split $x "."] 0]
    }
}
```

## Machine Cert

To allow machine certificate revocation validation, add a variable assign with 2 following variables before OSCP or CRLDP boxes.

### session.ssl.cert.whole

store machine certificate as it was user certificate

```
expr {[mcget {session.check_machinecert.last.cert.cert}]}
```

### session.ssl.cert.certissuer

store machine certificate issuer as it was user certificate issuer

```
expr {[mcget {session.check_machinecert.last.cert.issuer}]}
```

## HTTP auth returned cookie parsing

### session.custom.http\_auth\_mycookie

extract from HTTP auth cookie list the cookie value of mycookie

```
expr { [lindex [regexp -inline {mycookie=([^;\\r]*)} [mcget session.http.last.response_cookie]] 1] }
```

## replace portal or network access Webtop by full webtop if unsupported resource are assigned

Webtop can be:

- Portal webtop : define an internal web server as home page
- Network access Webtop : start automatically Network access when connected
- Full Webtop : display all assigned ressources in one page hosted on the F5.  
Some customers want to assign different webtop based on assigned ressources.
- one portal ressource only -> portal webtop
- one Network access ressource only -> Network Access ressource
- more than one portal ressource -> Full webtop
- more than one Network access ressource -> Full webtop
- RDP, Application tunnel, SAML ressources assigned -> Full Webtop

In Advanced ressource assign, the last assigned webtop is applied to the session. If the user is assigned non portal ressource (ex : RDP) and portal webtop, he will not be allowed to connect.

### session.assigned.webtop

this code is used if portal or network access webtop are assigned and number of resources is supported only with full webtop

```
set fullwt /Common/wt-Full;
set wt [mcget {session.assigned.webtop}];
set pa [llength [mcget {session.assigned.resources.pa}]];
set at [llength [mcget {session.assigned.resources.at}]];
set na [llength [mcget {session.assigned.resources.na}]];
set rd [llength [mcget {session.assigned.resources.rd}]];
set saml [llength [mcget {session.assigned.resources.saml}]];
if { $rd || $at || $saml || ([expr { $pa + $na }] > 1) } {set wt $fullwt};
unset fullwt;
```

```
unset pa;
unset at;
unset na;
unset rd;
unset saml;
return $wt;
```

one-line code. Don't forget to replace "/Common/wt-Full" with your own webtop full in expression.

```
expr { [length [concat [mcget {session.assigned.resources.rd}] [mcget {session.assigned.resources.at}]
[mcget {session.assigned.resources.atsaml}]]] || [length [concat [mcget {session.assigned.resources.pa}]
[mcget {session.assigned.resources.na}]]] >1 ? "/Common/wt-Full" : [mcget {session.assigned.webtop}] }
```

Same condition for Advanced resource Assign condition. This condition doesn't match with previous rules in the same Advanced resource assign. must be in a dedicated resource assign box.

```
expr { [length [concat [mcget {session.assigned.resources.rd}] [mcget {session.assigned.resources.at}]
[mcget {session.assigned.resources.atsaml}]]] || [length [concat [mcget {session.assigned.resources.pa}]
[mcget {session.assigned.resources.na}]]] >1 }
```

## For Kerberos SSO

when working with Kerberos SSO, 2 variable sources must be set:

- username : must be equal to user sAMAccountName
  - domain : must be equal to user FQDN domain
- When working on access policy with multiple SSO method depending on the URI, Host header or some other parameters, you may have conflict on default SSO variables.
- For example, for Exchange :
- activesync SSO profile is basic with username format is NTDOMAIN\username
  - Autodiscover SSP profile can be NTLM with
    - username format is username
    - domain format is NTDOMAIN
  - OWA SSO profile can be kerberos with
    - username : must be equal to user sAMAccountName
    - domain : must be equal to user FQDN domain like DOMAIN.LOCAL (different than NT Domain)
- default SSO variables are :
- session.sso.token.last.username
  - session.sso.token.last.password
  - session.logon.last.domain

to support multiple SSO on the same Access policy, I recommend to set new variables based on previous AD Query

## session.krbssso.username

```
expr {[mcget {session.ad.last.attr.sAMAccountName}]}
```

## session.krbssso.domain

```
expr {[mcget {session.ad.last.actualdomain}]}
```

# Links

Thank you for this: <https://devcentral.f5.com/s/articles/apm-variable-assign-examples-1107>

---

Revision #6

Created 17 March 2021 07:07:20

Updated 20 September 2022 09:58:29