Code-Examples

Perl

List domains of customer

```
#!/usr/bin/perl
use LWP::UserAgent;
use Data::Dumper;

my $dmapiURL = "https://dmapi.joker.com/request";

my $ua = LWP::UserAgent-> new;

my $req = HTTP::Request-> new(GET => $dmapiURL.'/login?username='.shift().'&password='.shift());
my $res = $ua-> request($req);

unless ($res-> is_success) {
print "Failed: ", $res-> status_line, "n";
exit -1;
}

my $auth=$res-> as_string;
my $output;

if ($auth =~ /Auth-Sid:s*(w+)/ms) {
$req=HTTP::Request-> new(GET => $dmapiURL.'/query-domain-list?auth-sid='.$1);
$res = $ua-> request($req);
if ($res-> is_success) {
$output=$res-> content;
}

else{
```

```
print "Failed: ", $res-> status_line, "n";
exit -2;
}
}

else {
print "Failed: Got no auth-id from DMAPI:n";
print $res-> content;
exit -3;
}



my @lines=split ("n",$output);
my $month=now + ("2M");
my %list;

my $p=0;
for my $line(@lines) {
#Skip first line
next unless $p or $line =~ /^s*$/;
unless ($p) { $p=1; next};
my ($fqdn,$exp)=split ("[ t]+",$line);
$list{$fqdn}=$exp;
}

for (sort keys %list) {
print "$_n";
}
```

List domains to expire next month

Needs:

Class::Date and Date::Parse from CPAN!

```
#!/usr/bin/perl

use LWP::UserAgent;
use Data::Dumper;
```

```
use Class::Date qw(:errors date localdate gmdate now -DateParse);

my $dmapiURL = "https://dmapi.joker.com/request";

my $ua = LWP::UserAgent-> new;
my $req = HTTP::Request-> new(GET => $dmapiURL. '/login?username='.shift().'&password='.shift());
my $res = $ua-> request($req);

unless ($res-> is_success) {
print "Failed: ", $res-> status_line, "n";
exit -1;
}

my $auth=$res-> as_string;
my $output;

if ($auth =~ /Auth-Sid:s*(w+)/ms) {
$req=HTTP::Request-> new(GET => $dmapiURL. '/query-domain-list?auth-sid='.$1);
$res = $ua-> request($req);

if ($res-> is_success) {
$output=$res-> content;
}
else {
print "Failed: ", $res-> status_line, "n";
exit -2;
}
}
else {
print "Failed: Got no auth-id from DMAPI:n";
print $res-> content;
exit -3;
}



my @lines=split ("n",$output);
my $month=now + ("2M");
my %list;
```

```
my $p=0;
for my $line(@lines) {
#Skip first line
next unless $p or $line =~ /^s*$/;
unless ($p) { $p=1; next};
my ($fqdn,$exp)=split ("[ t]+",$line);
next unless localdate($exp) < $month;
$list{$fqdn}=$exp;
}

for ( map {$_-> [0]}
sort {
$a-> [1] < => $b-> [1]
||
$a-> [2] cmp $b-> [2]
}
map {[$_, localdate $list{$_},$_]}
keys %list ) {
print "$_:".$list{$_}."n";
}
```

Replace admin-c in multiple domains

```
#
# replace admin-c in multiple domains
#
#!/usr/bin/perl
use LWP::UserAgent;
use Data::Dumper;
my $ua = LWP::UserAgent->new;
my $req = HTTP::Request->new(GET =>
'https://dmapi.joker.com/request/login?username='.shift().'&password='.shift());
my $res = $ua->request($req);
#--Admin-C :
my %ADMIN = ( de =>'CODE-12345' , org => 'CORG-12345', com => 'CCOM-12345', 'eu' => 'c12345' );
unless ($res->is_success) {
  print "Failed: ", $res->status_line, "\n";
  exit -1;
```

```
}
my $auth=$res->as_string;
my $output;
if ($auth =~ /Auth-Sid:\s*([a-z0-9]+)/m ) {
$auth = $1;
} else {
  print "Failed: Got no auth-id from DMAPI:\n";
  print $res->content;
  exit -3;
}
while ( <DATA> ) {
  chomp;
  my $fqdn = $_;
  ## for 3rd level domains this must be changed!
  my ( $sld, $tld ) = split ( /\./ , $fqdn );
  my $admin = $ADMIN{ $tld };
$req=HTTP::Request->new(GET => 'https://dmapi.joker.com/request/domain-modify?domain='.$fqdn.'&admin-
c='.$admin.'&auth-sid='.$auth);
$res = $ua->request($req);
  if ($res->is_success) {
$output=$res->content;
     print $output;
  } else {
     print "Failed: ", $res->status_line, "\n";
     exit -2;
  }
}
__DATA__
domain1.de
domain2.eu
domain3.org
domain4.com
__END__
```

PHP

Introduction

Please note that there is a full DMAPI based interface written in PHP available. It is free to download and modify, and could be downloaded from Github.

Please read more at section Joker PHP client.

The published examples in PHP are parts of this project and are stripped in terms of functionality and error handling. It is strongly recommended to use the complete code base from Github.

List domains of a customer

```
<?php

//sends HTTP request using CURL

function query_host($conn_server, $params = "", $get_header = false)
{

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $conn_server.$params);
    if (preg_match("/^https:\/\//i", $conn_server)) {
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
    }

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    if ($get_header) {
        curl_setopt($ch, CURLOPT_HEADER, 1);
    }

    else {
        curl_setopt($ch, CURLOPT_HEADER, 0);
    }

    $result = curl_exec($ch);

    if (curl_errno($ch)) {
        print "curl error";
```

```
    }

    else {
        curl_close($ch);
    }

    return $result;
}

//builds query, sends request and gets the answer back

function execute_request($request, $params, &$sessid)
{
    //build the query
    $http_query = "/request/" . $request . "?" . $params   ."&auth-sid=".$sessid."&client-
ip=".$_SERVER["REMOTE_ADDR"];
    //send the request
    $raw_res = query_host("https://dmapi.joker.com", $http_query, true);
    $temp_arr = @explode("\r\n\r\n", $raw_res, 2);
    //split the response for further processing
    if (is_array($temp_arr) && 2 == count($temp_arr)) {
        return $temp_arr[1];
    }

    else {
        return false;
    }
}

//basic parsing of the DMAPI header

function parse_response_header($header)
{
    $raw_arr = explode("\n", trim($header));
    $result = array();
    if (is_array($raw_arr)) {
        foreach ($raw_arr as $key => $value)
        {
            $keyval = array();
```

```
            if (preg_match("/^([^\s]+):\s+(.+)\s*$/", $value, $keyval)) {
                $arr[strtolower($keyval[1])] = $keyval[2];
            }

            else {
                print "Header line not parseable - pattern does not match\nRaw header:\n$value";
            }
        }
    }

    else {
        $arr = "";
        print "Unidentified error\nRaw header:\n$header";
    }

    return $arr;
}

//parses the reply from DMAPI into a header and body
function parse_response($res)
{
    $raw_arr = explode("\n\n", trim($res));
    $arr_elements = count($raw_arr);
    if ($arr_elements > 0) {
        $temp["response_header"] = parse_response_header($raw_arr["0"]);
        $temp["response_body"] = $raw_arr["1"];
    }

    else {
        print "Couldn't split the response into response header and response body\nRaw result:\n$res";
        $temp = "";
    }

return $temp;
}

$response = "";
$authid = "none";
//first obtaining an Auth-ID
```

```
$username = "<your username here>";
$password = "<your password here>";
$fields = "username=".urlencode($username)."&password=".urlencode($password);
$result = execute_request("login", $fields, $authid);
$result = parse_response($result);
$authid = $result["response_header"]["auth-sid"];
if (!$authid) {
    print "no auth-id was obtained - probably due to wrong username or password";
    exit;
}

//now asking for the domain list
$pattern = "*";
$fields = "pattern=".urlencode($pattern);
$result = execute_request("query-domain-list", $fields, $authid);
$result = parse_response($result);

print_r($result["response_header"]);
print "< br /> ";
print_r($result["response_body"]);
```

Python

Introduction

Python is easy to use with DMAPI. Our examples are tested with python 3 on Windows and Linux, but should work on all platforms.

Running the Python example will look like:

```
$ ./dmapi-example.py
Request-URL:  https://dmapi.ote.joker.com/request/login
Login: Status-Code: 0

Request-URL:  https://dmapi.ote.joker.com/request/query-domain-list
Domain List: Status-Code: 0
```

```
 domain: another-privacy-test.com
 expiration_date: 2018-06-30

 domain: another-privacy-test.net
 expiration_date: 2020-06-30

Request-URL:  https://dmapi.ote.joker.com/request/logout
Logout: Status-Code: 0
```

Login and list your domains

```
#!/usr/bin/env python
import requests

dmapiURL = 'https://dmapi.ote.joker.com'
dmapiUser = 'username'
dmapiPassword = 'password'

def main():
    loginResponse = login(dmapiUser,dmapiPassword)
    print("Login: Status-Code:", loginResponse.header['Status-Code'])
    if loginResponse.header['Status-Code'] !='0':
        print(loginResponse.header['Status-Text'])
        return

    sessionId = loginResponse.header['Auth-Sid'];
    print("")
    domainResponse = domainList(sessionId,1,5)
    print("Domain List: Status-Code:", domainResponse.header['Status-Code'])
    print("")
    domains = domainResponse.resultListWithNames()
    for domain in domains:
        for key, value in domain.items():
            print(" %s: %s" % (key, value))
        print("")
    logoutResponse = logout(sessionId)
    print("Logout: Status-Code:", logoutResponse.header['Status-Code'])
```

```
# implement dmapi commands as functions
def login(username,password):
    parameters = { 'username': username, 'password': password }
    message = sendCommand('login', parameters)
    return message;

def logout(sessionId):
    parameters = { 'auth-sid': sessionId }
    message = sendCommand('logout', parameters)
    return message;

def domainList(sessionId, list_from=1, list_to=""):
    parameters = { 'auth-sid': sessionId , 'from': list_from, 'to': list_to }
    message = sendCommand('query-domain-list', parameters)
    return message;

# general dmapi command call
def sendCommand(command,parameter={}):
    try:
        url = dmapiURL+'/request/'+command
        print("Request-URL: ", url)
        response = requests.get(url, params=parameter)
        # print URL with parameters for debugging purposes
        # print("Request-URL: ", response.url)
        if response.status_code != requests.codes.ok:
            raise CommandError("Command Failed! HTTP Status Code: %s" % response.status_code)
        return DmapiResponse(response.text)
    except requests.ConnectionError as e:
        raise CommandError("Connection Error: %s" % str(e))
    except requests.HTTPError as e:
        raise CommandError("Http Error: %s" % str(e))
    except CommandError as e:
        raise
    except Exception as e:
       raise CommandError("Unexpected Error: %s" % str(e))

class DmapiResponse():
```

```
def __init__(self,responseBody):
    parts = responseBody.split("\n\n",1)
    if len(parts)>0:
        self.header = self.__parseKeyValueList(parts[0])
    if len(parts)>1:
        self.body = parts[1]

def __parseKeyValueList(self,text):
    lines = text.split("\n")
    keyValueList = {}
    for line in lines:
        keyValue = line.split(' ',1)
        key = keyValue[0].rstrip(':')
        value = keyValue[1]
        keyValueList[key] = value
    return keyValueList

def __getSeparator(self):
    if self.header.get('Separator') == 'TAB':
        return "\t"
    else:
        return " "

def resultList(self):
    lines = self.body.split("\n")
    resultList = []
    separator = self.__getSeparator()
    for line in lines:
        values = line.split(separator)
        resultList.append(line.split(separator))
    return resultList

def resultListWithNames(self):
    columnNames = self.resultListColumns()
    resultList = []
    if len(columnNames) > 0:
        rawList = self.resultList()
        resultList = []
        for row in rawList:
```

```
            columns = {}
            for idx, column in enumerate(row):
                columns[columnNames[idx]] = column
            resultList.append(columns)
        return resultList

    def resultListColumns(self):
        if 'Columns' in self.header:
            columnsText = self.header['Columns']
            columns = columnsText.split(',')
            return columns
        else:
            return []

    def resultValues(self):
        return self.__parseKeyValueList(self.body)

class CommandError(Exception):
    def __init__(self, value):
        self.value = value
    def __str__(self):
        return repr(self.value)

# call main function
try:
    main()
except CommandError as e:
    print("Error:", str(e).strip("'"))
```

Login and list all A and CNAME records

```
#!/usr/bin/env python
import requests

dmapiURL = 'https://dmapi.joker.com'
dmapiUser = 'username'
dmapiPassword = 'password'
```

```
def main():
    loginResponse = login(dmapiUser,dmapiPassword)
    #print("Login: Status-Code:", loginResponse.header['Status-Code'])
    if loginResponse.header['Status-Code'] !='0':
        print(loginResponse.header['Status-Text'])
        return

    sessionId = loginResponse.header['Auth-Sid'];
    #print("")
    dnsZoneListResponse = dnsZoneList(sessionId,"")
    #print("DNS Zone List: Status-Code:", dnsZoneListResponse.header['Status-Code'])
    dnslist = dnsZoneListResponse.resultList()
    for row in dnslist:
        domain = row[0]
        expiration = row[1]
        #print(" domain: %s" % (domain))
        #print(" expiration: %s" % (expiration))
        #print("")
        dnsZoneResponse = dnsZoneGet(sessionId, domain)
        #print("DNS Zone GET for %s: Status-Code:" % (domain), dnsZoneResponse.header['Status-Code'])
        zoneEntries = dnsZoneResponse.resultList()
        for entry in zoneEntries:
            #print(' '.join(entry))
            if len(entry)<5:
                continue
            eLabel = entry[0]
            eType = entry[1]
            ePriority = entry[2]
            eTarget = entry[3]
            eTTL = entry[4]
            if eType == 'A' or eType == 'CNAME':
                print(("%s.%s\t%s" % (eLabel,domain,eTarget)).lstrip('@.'))

    logoutResponse = logout(sessionId)
    #print "Logout: Status-Code:", logoutResponse.header['Status-Code']



# implement dmapi commands as functions
def login(username,password):
```

```
    parameters = { 'username': username, 'password': password }
    message = sendCommand('login', parameters)
    return message;

def logout(sessionId):
    parameters = { 'auth-sid': sessionId }
    message = sendCommand('logout', parameters)
    return message;

def domainList(sessionId, pattern="", list_from=1, list_to=""):
    parameters = { 'auth-sid': sessionId , 'from': list_from, 'to': list_to, 'pattern': pattern }
    message = sendCommand('query-domain-list', parameters)
    return message;

def dnsZoneList(sessionId, pattern="", list_from=1, list_to=""):
    parameters = { 'auth-sid': sessionId , 'from': list_from, 'to': list_to, 'pattern': pattern }
    message = sendCommand('dns-zone-list', parameters)
    return message;

def dnsZoneGet(sessionId, domain):
    parameters = { 'auth-sid': sessionId , 'domain': domain }
    message = sendCommand('dns-zone-get', parameters)
    return message;

# general dmapi command call
def sendCommand(command,parameter={}):
    try:
        url = dmapiURL+'/request/'+command
        #print("Request-URL: ", url)
        response = requests.get(url, params=parameter)
        # print URL with parameters for debugging purposes
        # print("Request-URL: ", response.url)
        if response.status_code != requests.codes.ok:
            raise CommandError("Command Failed! HTTP Status Code: %s" % response.status_code)
        return DmapiResponse(response.text)
    except requests.ConnectionError as e:
        raise CommandError("Connection Error: %s" % str(e))
    except requests.HTTPError as e:
        raise CommandError("Http Error: %s" % str(e))
```

```
    except CommandError as e:
        raise
    except Exception as e:
       raise CommandError("Unexpected Error: %s" % str(e))

class DmapiResponse():
    def __init__(self,responseBody):
        parts = responseBody.split("\n\n",1)
        if len(parts)>0:
            self.header = self.__parseKeyValueList(parts[0])
        if len(parts)>1:
            self.body = parts[1]

    def __parseKeyValueList(self,text):
        lines = text.split("\n")
        keyValueList = {}
        for line in lines:
            keyValue = line.split(' ',1)
            key = keyValue[0].rstrip(':')
            value = keyValue[1]
            keyValueList[key] = value
        return keyValueList

    def __getSeparator(self):
        if self.header.get('Separator') == 'TAB':
            return "\t"
        else:
            return " "

    def resultList(self):
        lines = self.body.split("\n")
        resultList = []
        separator = self.__getSeparator()
        for line in lines:
            values = line.split(separator)
            resultList.append(line.split(separator))
        return resultList

    def resultListWithNames(self):
```

```
        columnNames = self.resultListColumns()
        resultList = []
        if len(columnNames) > 0:
            rawList = self.resultList()
            resultList = []
            for row in rawList:
                columns = {}
                for idx, column in enumerate(row):
                    columns[columnNames[idx]] = column
                resultList.append(columns)
        return resultList

    def resultListColumns(self):
        if 'Columns' in self.header:
            columnsText = self.header['Columns']
            columns = columnsText.split(',')
            return columns
        else:
            return []

    def resultValues(self):
        return self.__parseKeyValueList(self.body)

class CommandError(Exception):
    def __init__(self, value):
        self.value = value
    def __str__(self):
        return repr(self.value)

# call main function
try:
    main()
except CommandError as e:
    print("Error:", str(e).strip("'"))
```

C#

Applications for domain management with DMAPI are very easy to create using MS Visual Studio. This example code logs into the Joker.com system using DMAPI requests.

It provides a domain listing, whois lookup and "Email verification".

This has been tested using Visual Studio 2008 and 2010, but should work with any version, and also with free Visual Studio Express and .NET versions from 3.5 onwards. Please find the source code as an attachment below.

We also provide this as a windows binary to try without the need of building it before. In case you run Windows 10 with "Smartscreen", you have to click on "more information" to execute this, since this binray of course is not signed by a "manufacturer"...

This is how this will look like:

C#/.NEt DMAPI Example

Revision #18
Created 14 September 2023 11:17:46 by Admin
Updated 2 October 2024 09:37:35 by Administrator