BlockIP2 The manual

Further input/tips is appreciated. Feel free to drop us a email.

Quick Links

Foreword

It have always been a challenge to keep our WebSphere MQ Networks secure. Today in WebSphere MQ version 5.3 IBM is offering us the ability to use SSL to prevent intruders. Anyway SSL requires a lot of administration and creating an infrastructure that supports SSL, CA etc. When I started working with WebSphere MQ we only had it as an internal tool, and there was no client connections allowed. That was an manageable solution. After the introduction of MQ-clusters and the usage of Client Connections it became a challenge to keep the WebSphere MQ network secure.

This was the trigger to start building a simple WebSphere MQ Security exit that was able to help me keep our networks secure, I had to do a lot studying the intercommunication guide, I did a password verification exit, similar to CSQ4BCX3 (supplied together with MQ 5.3.1 for z/OS). When this exit was finished I began to look on the security, and saw that we also had to do some IP-filtering, and I needed the ability to log connection attempts, and sometime had to do investigation when spammers entered the network, and was trying to connect, using a channel. This lead to development of the LogIP and BlockIP exits.

When the exits was released late 2002, I could see that it was an success, I had some chats on my first visit on the T&M conference in Las Vegas in 2003, and I could feel that there was a need to promote the exit, because it was the same situation all WebSphere MQ administrators was facing: Keep MQ tight.

What started just as an simple exit have now been ported to z/OS, Linux, Windows, HP-UX, Sun/Solaris. A lot of specialists round the globe have contributed to BlockIP2 with requirements, coding, testing etc.

The new MAXCHL connection control was in the beginning based on the Support pack

ME71 written by Tony Madden, MQWare UK. A sort of this implementation lives in the Windows version. Under z/OS and UNIX/Linux I've moved into shared memory, to gain high performance without a high CPU consumption.

Top of page

Installation

When you have downloaded the BlockIP2.zip and unpacked the files it's time to install the Exit.

Windows 2000/2003/NT

Just copy the BlockIP2.dll and BlockIP2S.exe into the "exits" directory where WebSphere MQ is installed, this makes it very easy to configure the exit. It's VERY important that BlockIP2S.exe is placed in your "exits" directory. This program is used to establish shared memory for BlockIP2.

z/OS

You have to upload the blockip.mvs (containing BlockIP2.c), blockip2.jcl.txt and blockipw.txt.

I have added the BlockIP2.c in Xmit format ready to upload, without having to deal with the ASCII to EBCDIC conversion, you just upload it to z/OS using this ftp command:

ftp -s:upload.txt hostname (Just remember to change <userid> and <password> to yours before using it, or use another way to upload the file binary to a PS with a fixed record length of 80).

```
<userid>
<password>

bin
quote SITE recfm=fb lrecl=80 blksize=8000
put blockip2.mvs
put blockip2.mvso
ascii
put blockipw.txt
quit
```

example upload.txt

When you have uploaded **blockip2.mvs** and **blockip2.mvso** you have to issue a receive to get into a usable dataset:

TSO RECEIVE INDATASET(BLOCKIP2.MVS)
TSO RECEIVE INDATASET(BLOCKIP2.MVSO)

This will give you a PDS with a LRECL=255 and RECFM=VB

When you successfully have uploaded the stuff, it's time to assemble and compile but before you're able to do that you have to change the JCL to match your installation. First submit blockipw, to create the WTO routine. When this is done submit blockip2.jcl.txt. When this is run ok, you should have a working security exit.

On your xxxxCHIN task add a //SYSPRINT DD card to catch output from BlockIP2, and you can add a DD card to handle the configuration file. The link-edited modules must be placed in the data set specified by the CSQXLIB DD statement of the channel initiator address space procedure; the names of the load modules are specified as the exit names in the channel definition.

The exits are invoked as if by an z/OS LINK, in:

- Non-authorized problem program state
- Primary address space control mode
- Non-cross-memory mode
- Non-access register mode
- 31-bit addressing mode

I don't distribute (and don't plan to) a z/OS load module due to the complexity of your different implementations, software levels, requirements etc.

You can however upload BlockIP2.c to z/OS using FTP and handle ASCII to EBCDIC conversion like this (just remember to create a dataset named 'MQ.BCLOCKIP.C, the a LRECL=255 and RECFM=VB:

```
<userid>
<password>


cd 'MQ.BLOCKIP.C'
quote SITE SBDataconn=(IBM-1047,ISO8859-1)
put blockip2.c
quit
```

myftpparm.txt

ftp -s:myftpparm.txt hostname

The trick is done by **`quote SITE SBDataconn=(IBM-1047,ISO8859-1)`** which tells z/OS to do the conversion. The codepages IBM-1047 and ISO8859-1 differs depending on where you are located, so talk to you z/OS Unix Systems Service administrator to obtain the correct values in your situation. This conversion can also be used when you download a coded file from z/OS.

Linux (Intel 32 bit)

Just untar BLOCKIP2.TAR from Linux_x86 subdir in /var/mqm/exits, and you're ready to go.

Linux (Intel 64 bit)

Just untar BLOCKIP2.TAR from Linux_x86_64 subdir in /var/mqm/exits64, and you're ready to go.

AIX

Just untar BLOCKIP2.TAR from AIX subdir in /var/mqm/exits, and you're ready to go.

Sun/Solaris

You have to compile the exit your self, because I don't have this platform.

Sun/Solaris (x86_64)

Just untar BLOCKIP2.TAR from Solarisx86 subdir in /var/mqm/exits64, and you're ready to go.

HP-UX

You have to compile the exit your self, because I don't have this platform.

Top of page

Activation

You specify that WebSphere MQ have to use the security exit by using the SCYEXIT and SCYDATA keywords when defining/altering channels. This could be like this:

```
ALTER CHANNEL(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('172.20.10.*;172.221.*') +
  scyexit('BlockIP2(BlockExit)')
```

This works on Linux, Windows.

If it was on z/OS it would look like this:

```
ALTER CHANNEL(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('172.20.10.*;172.221.*') +
  scyexit('BLOCKIP2')
```

For further information on the commands refer to the MASC. and Intercommunication guides.

Top of page

Configuration

As you saw in the introduction, you control the behavior of BlockIP2 using SCYDATA().

You can specify a simple pattern, or use keywords.

The syntax is:

SCYDATA('{pattern[;pattern[;pattern...]][;userids][;flags]}|FN=filename;[flags;]}')

A simple filtering

SCYDATA('172.20.*') to allow anybody from the network 172.20.anything to access the queue manager.

Yet another SCYDATA('172.??.10.21') this one allows only connections from 172.10.10.21 172.11.10.21, it requires two digits in second IP byte.

Flags implemented

Flag	Description
-d	Debugging activated
-z	z/OS only: Translate [] to <> on messages
-n	Block system accounts: mqm, musr_mqadmin and "" (unspecified)
+b	Allow blank userids (unspecified) to connect. (added in version 2.21)
-q	quiet mode

To enable the debugging you specify:

SCYDATA('172.20.*;**-d;**')

Using a specification file (the way of specifying the file differs from platform to platform):

On windows:
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('**FN=**c:\path\Blockspec.txt;') +
  scyexit('BlockIP2(BlockExit)')

On Linux:
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('**FN=**/var/mqm/exits/Blockspec.txt;') +
  scyexit('BlockIP2(BlockExit)')

On AIX:
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +

```
  SCYDATA('FN=/var/mqm/exits/Blockspec.txt;') +
  scyexit('BlockIP2(BlockExit)')
```

On HP-UX:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('FN=/var/mqm/exits/Blockspec.txt;') +
  scyexit('BLOCKIP2')
```

On SUN/Solaris x86:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('FN=/var/mqm/exits/Blockspec.txt;') +
  scyexit('BlockIP2(BlockExit)')
```

On z/OS:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('FN=//DD:BLOCKDD;') +
  scyexit('BLOCKIP2')
```

Or if you're using a PDS, you can specify the member name aswell:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('FN=//DD:BLOCKDD(MEM);') +
  scyexit('BLOCKIP2')
```

Or using a USS file placed in HFS:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('FN=/var/mqm/exits/Blockspec;') +
  scyexit('BLOCKIP2')
```

Now what can we specify in the configuration file??

Top of page

Configuration file

Following keywords are implemented (version 2.21):

Keywords	**Description**
Patterns=<generic_connection_list>;	List of positive network connection names using generic pattern matching
Userids=<generic_user_list>;	List of positive userids using generic pattern matching
AllowBlankUserID=Y;	Allow blank userids to connect. **To Enable this option can lead to security risks.**
	Flag to block for system

BlockMqmUsers=Y;	accounts: **mqm, musr_mqadmin** and "
SSL=DN=<generic_DN;[{BLOCK|MCA=userid};]	fine secetion based on the DN, which also allow override of MCAUSERID and even to blacklist one or more DN.
CON=<conname>;<userids>[;MCA={*|userid}}|BLOCK}];	Allow manipulation/selection on specified connectionname/userid. This have higher precedence than userid/pattern filtering.
UseridUpperLowerCase=*;	Swift off case sensetivity, fold any matching to uppercase.
BlockUsers=<generic_user_list>;	List of negative userids using generic pattern matching
MAXCHL=<generic_ChannelName>;<numberOfChannels>;	Limit the number of connection to this channel, the number of connections are checked in the INIT phase.
LogFileName=<spec>;	Name of the logfile default is 'BlockLog'
LogExt=<spec>;	extension of logfile, default is 'txt'
LogFormat=<spec>;	N - For name only or ND - Name and Date or NDC - Name Date Channel format NCD - Name Channel Date format or any variation
LogPath=<spec>;	The path where the log file is placed
LogDrive=<spec>;	The drive there the logfile is placed **(windows only)**

Patterns=, Users= and BlockUsers= offers concatenation functionality.

This allows file content like:
Patterns=1.2.3.4,1.2.3.5,1.2.3.6; /* hosts 4,5,6 in the 1.2.3 network */
Patterns=1.2.1.4,1.2.1.5,1.2.1.6; /* hosts 4,5,6 in the 1.2.1 network */
and any of the six addresses will be allowed. this can also be specified as (after v.2.21):
Patterns=1.2.3.[4-6],1.2.1.[4-6];

Specifying pattern rules are described in details here

There is limit on 256 characters maximum length on Patterns, Users, BlockUsers.

There are room for 16 SSL patterns and 64 CON patterns. This limits can easily elevated, but it require more memory. The memory obtained by BlockIP2 remains allocated until the connection is terminated, therefore have I decided to set some limitation on.

Top of page

PWDConfiguration examples

To use multipattern: just seperate the patterns with a semicolon(;) like this:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
    SCYDATA('172.20.109.*;172.221.*;10.31.*') +
    scyexit('BlockIP2(BlockExit)') *
```

This will allow communication from any computer in the 172.20.109.*, 172.221.* and 10.31.* networks.

You can also use a single position placeholder in the pattern:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
   SCYDATA('192.168.??.20;10.31.*') +
   scyexit('BlockIP2(BlockExit)')
```

This will allow any IP-addr matching 192.168.10.20, 192.168.11.20.. 192.168.99.20 and 10.31.* to pass verification.

How to block bad userids (mqm, musr_mqadmin and ''), this is done using the switch **-n** in SCYDATA. When the RemoteUserIdentifier of an incoming connection request contains one of the listed users above BlockIP2 sends MQXCC_SUPPRESS_FUNCTION so the communication attempt is abandoned.

BlockIP2 have been updated in SCYDATA like this:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('172.20.10.*;172.221.*;10.31.*;-n') +
  scyexit('BlockIP2(BlockExit)')
```

A allow blank userid (unknown) to connect, this is done with the **+b** option.

```
alt chl(JMSCHL_SPY) chltype(SVRCONN) +
   SCYDATA('172.20.10.*;172.221.*;+b') +
   scyexit('BlockIP2(BlockExit)')
```

WARNING: Using this option is causing a security hole, so anybody using a java application or a security client exit to connect to the queue manager and get up to mqm authority.

A quiet mode option have been added, so there will be only one line per connection attempt in the log. This is done with the **-q** option.

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('172.20.10.*;172.221.*;-q') +
  scyexit('BlockIP2(BlockExit)')
```

A debug option is added, to allow a more comprehensive logging of the activity inside BlockIP2. This done with the **-d** option.

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
  SCYDATA('172.20.10.*;172.221.*;-d') +
  scyexit('BlockIP2(BlockExit)')
```

BlockIP2 is able to obtain the security specification from a file. This feature was added to circumvent the 32 byte limitation in SCYDATA, and was requested from complex installations.

The filename is specified in SCYDATA like this on **windows**:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
 SCYDATA('FN=c:\path..\Blockspec.txt;') +
 scyexit('BlockIP2(BlockExit)')
```

or like this on **Linux and the rest of unix impl**.:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
 SCYDATA('FN=/var/mqm/exits/Blockspec.txt;') +
 scyexit('BlockIP2(BlockExit)')
```

or like this on **z/OS**:

```
alt chl(MQT2.TCP.MQT1) chltype(SVRCONN) +
 SCYDATA('FN=DD:BLOCKDD;') +
 scyexit('BLOCKIP2')
```

NB: The **FN=**<filename>**;** requires the ending semicolon(;).

Here is a simple commented specification

```
# This is a comment BlockIP2 security specification Blockspec.txt
Patterns=10.1.*,172.20.31.*,127.0.?.1;
Userids=xxx,yyy,zzz*,etc,mrmq,root,us???mq;
BlockMqmUsers=Y;
#-----------------------------------------------------------------
# Description of security specification:
# This specification will allow:
#  - connections from 10.1.*, 172.20.31, and 127.0.0.1, 127.0.1.1, ... 127.0.9.1
#    Networks.  Entries are separated using comma(,)
#  - allow  userids: xxx, yyy, etc, mqm and mrmq.  Together with users
# starting  with zzz<something> and users beginning with us<three chars>mq.
#
# - BlockMqmUsers=Y; means that following users are blocked: mqm,
#    MUSR_MQADMIN and musr_mqadmin.
#  All specs must end with a semicolon ; anything after ; is parsed as a comment
# Important:
# If you specify Patterns twice they are concatenated, so they are all used
# for matching.
#
# It's very important that all specs are terminated with semicolon(;) otherwise
# you will receive connection refused. Because I think it's best only to use
# positive error-free identification
# The patterns are separated using comma(,) This is also important to remember
```

BlockIP2 is able to change MCAUSER depending on the incoming connection credentials.

```
# This is a comment BlockIP34.txt
Patterns=10.1.*,10.1.11.*,etc,127.0.?.1;
SSL=CN=ibmwebspheremqclient2;MCA=mrmq; ok userid
SSL=CN=ibmwebspheremqQM2;BLOCK; blocked user
SSL=CN=ibmwebspheremq*;MCA=*; just do nothing
SSL=CN=*;MCA=NoBody; Set all other to NoBody
# EOF
```

In the example above will connections with a CN=ibmwebspheremqclient2 get the MCAUSER=mrmq.

Connections from CN=ibmwebspheremqQM2 will be refused all other connections from CN=ibmwebspheremq* will use the normal userid (or MCAUSER if specified on the channel).

Connections made with CN=<anything> will have the MCAUSER=NoBody, which should be no defined. (Could also be SSL=CN=*;BLOCK; have the same function when user NoBody is undefined)

Here is an commented example using **CON=** (not from the real world), but still informative:

```
#
# Simple filter implemented in BlockIP2 version 2.20
#
# 1. stop all connection attempts from mqm (NoBody is an undefined or blocked userid).
CON=*;mqm;MCA=NoBody;
#
# 2. Stop users starting with dk14 from 10.31.* Might be a foregin network
CON=10.31.*;dk14*;MCA=NoBody;
#
# 3. Allow master03 when comming from 172.20.10.31
CON=172.20.10.31;master03;
#
# 4. Allow spider when comming from 10.*, and set MCAUSER to master04
CON=10.*;spider;MCA=master04;
```

```
#
# 5. Block all other attempts.
CON=*;*;MCA=NoBody;
#
# Control the number of connections:
#
# Establish a default max connection count of 5
MAXCHL=*;5;
#
# Specific channel limitations
MAXCHL=SYSTEM.DEF.SVRCONN;25;
MAXCHL=SYSTEM.ADMIN.SVRCONN;25;
MAXCHL=SYSTEM.AUTO.SVRCONN;25;
```

The list is searched from top to bottom, and when BlockIP2 detects a match on connection-id and userid, this means that it's important to specify the rules in the right order.

Caution: Using both SSL and CON keywords in the same specification file, should be done with great caution, because the SSL have precedence over CON. You might see that a MCA change caused by CON, can affect a SSL setting without MCA settings.

Here is an commented example using **MAXCHL=** (not from the real world), but still informative:

```
#
# Control the number of connections:
#
# Establish a default max connection count of 5
MAXCHL=*;5;
#
# Specific channel limitations
MAXCHL=SYSTEM.DEF.SVRCONN;25;             Allow 25 connections here
MAXCHL=SYSTEM.ADMIN.SVRCONN;25;             Allow 25 connections here
MAXCHL=SYSTEM.AUTO.SVRCONN;25;              Allow 25 connections here
```

This feature implemented using shared memory on UNIX, Linux and z/OS. On windows it's implemented using "Display Channel Status" and a local queue: **SYSTEM.ADMIN.BIP2.QUEUE.**

You create the control queue like this:
DEFINE QL('SYSTEM.ADMIN.BIP2.QUEUE') defperst(NO) maxqdepth(500) replace +
DESCR('Blockip2 control queue')

The MAXCHL= table is processed sequential, and all matches are processed, this means that the last match have higher priority than the first one. Therefore should you place the default value as the first entry, and place specific entries after it.

BlockIP2 rely on /var/mqm/qmgrs/QMGR/shmem/BlockIP2.1

and /var/mqm/qmgrs/QMGR/shmem/BlockIP2.2 for pointing out the shared memory segment in the UNIX implementation.

Top of page

Specifying patterns

Special character	Rule description
*	Any number of characters towards end of spec
?	Any character in this position
#	Any numeric (0-9) character in this position
$	Any alphabetic character a-z and A-Z
(0-4) (c-j) (A-z)	Any alphanumeric character in the specified range (Added for european z/os systems)
[0-4] [c-j] [A-z]	Any alphanumeric character in the specified range

> **Remark:** The range selection is based on the character value, this means that there are a huge difference between z/OS, AS/400 and the other platforms because z/OS and AS/400 are using EBCDIC and the rest are using ASCII, this means that [0-z] will allow any character on Linux, but is a bad specification on z/OS. There are currently (v.2.21) added a z/OS limitation that are blocking for mixing numbers and letters in a range spec.

Here are some examples of how to specify various patterns :

```
#
# Simple filter implemented in BlockIP2 version 2.21
#
# 1. using range in connection name
# This statement allow connections from connection names that are starting with 10.1 having 2-5 in pos 5, having 0-9 in 7'th
# position and anything in the last group (0-255). This also means that you can't get in from 10.12.10.11 ...
# Net we're going to look on the userid, it must start with the 'us' and 6 digits, like us0234237, and you'll not be able to
# connect if you user id us02345 (one digit too short.
CON=10.1[2-5].#.*;us######;MCA=american;
#
# 2. using alphabetic filter in userid
# This statement allows connections from 172.20.10.21 only, But the userid have to start with the letters 'gb' followed by
#  three letters and ending with two zeroes '00'. This means that the userids would look like: gbabb00, gbwho00.
# And gbilo44 is not allowed.
CON=172.20.10.21;gb$$$00;MCA=uksys;
#
```

```
# 3. using range in userid
# This statement allows connections from 172.20.*, But the userid have to start with the letter 'd' followed by a letter
# in the range from 'a' thru 'd' followed by '00' and a letter in the range 'w' thru 'z'. This means that the userids would look
# like: da00w, dd00z. And de00a is not allowed.
CON=172.20.*;d(a-d)00(w-z);MCA=agent007;
#
# 4. Block all other attempts.
CON=*;*;MCA=NoBody;
#
```

Top of page

Using LogFormat

LogFormat offers you the ability to generate various logfiles, based on date, channelname, name.

Options available:
LogFormat=
N - For name only or
ND - Name and Date or
NDC - Name Date Channel format
NCD - Name Channel Date format or any variation

This option gives you the ability to generate a new file each day, which makes it easy to investigate who did what on a certain time.

WARNING: Don't use this option on z/OS

Top of page

Logfile example

Example of the log: **<will be replaced with current>**

```
04.01.20 00:11:04 BlockIP2: BlockExit ver=1.112 env=non-MVS ExitId=11 ExitReason=16 ChannelType=7
04.01.20 00:11:04 BlockIP2: BlockExit QMgr="QSJHPT01" ChannelName="CHANNEL2" ConnectionName="127.0.0.1" Userid="mrmq"
04.01.20 00:11:04 BlockIP2: ConName"127.0.0.1" Pattern "127.0.0.*;172.221.*;10.31.*;" Flags"-n ".
04.01.20 00:11:04 BlockIP2: BlockExit Connection accepted for pattern 127.0.0.* .
04.01.20 00:11:28 BlockIP2: BlockExit ver=1.112 env=non-MVS ExitId=11 ExitReason=16 ChannelType=7
04.01.20 00:11:28 BlockIP2: BlockExit QMgr="QSJHPT01" ChannelName="CHANNEL3" ConnectionName="127.0.0.1" Userid="mrmq"
04.01.20 00:11:28 BlockIP2: ConName"127.0.0.1" Pattern "127.0.0.2" Flags"".
04.01.20 00:11:28 BlockIP2: BlockExit Connection refused.
04.01.20 00:15:33 BlockIP2: BlockExit ver=1.112 env=non-MVS ExitId=11 ExitReason=16 ChannelType=7
04.01.20 00:15:33 BlockIP2: BlockExit QMgr="QSJHPT01" ChannelName="CHANNEL2" ConnectionName="127.0.0.1" Userid="musr_mqadmin"
04.01.20 00:15:33 BlockIP2: BlockExit Connection refused for system user identifier.-3
04.01.20 00:16:23 BlockIP2: BlockExit ver=1.112 env=non-MVS ExitId=11 ExitReason=16 ChannelType=7
04.01.20 00:16:23 BlockIP2: BlockExit QMgr="QSJHPT01" ChannelName="CHANNEL2" ConnectionName="127.0.0.1" Userid=""
```

```
04.01.20 00:16:23 BlockIP2: BlockExit Connection refused for blank user identifier.
04.01.22 23:22:45 BlockIP2: BlockExit ver=1.12 env=non-MVS ExitId=11 ExitReason=16 ChannelType=7
04.01.22 23:22:45 BlockIP2: BlockExit QMgr="QSJHPT01" ChannelName="JHP" ConnectionName="127.0.0.1" Userid="root"
04.01.22 23:22:45 BlockIP2: ConName"127.0.0.1" Pattern "128*" Flags "".
04.01.22 23:22:45 BlockIP2: BlockExit Connection refused for Conname 127.0.0.1 .
04.01.22 23:23:07 BlockIP2: BlockExit ver=1.12 env=non-MVS ExitId=11 ExitReason=16 ChannelType=7
04.01.22 23:23:07 BlockIP2: BlockExit QMgr="QSJHPT01" ChannelName="CHANNEL2" ConnectionName="127.0.0.1" Userid="root"
04.01.22 23:23:07 BlockIP2: ConName"127.0.0.1" Pattern "128*;127.*;" Flags "-n".
04.01.22 23:23:07 BlockIP2: BlockExit Connection accepted for pattern 127.* .
04.02.15 09:07:17 BlockIP2: BlockExit QMgr="QSJHPT01" ChannelName="CHANNEL2" ConnectionName="127.0.0.1" Userid="root"
04.02.15 09:07:17 BlockIP2: BlockExit Connection refused, invalid file name pattern specified: FN=<filename...>;
"FN=/var/mqm/exit/blockip2.txt " .04.02.15 09:08:08 BlockIP2: BlockExit QMgr="QSJHPT99" ChannelName="JHP"
ConnectionName="127.0.0.1" Userid="root"
04.02.15 09:08:08 BlockIP2: BlockExit Connection refused, Specified file "/var/mqm/exit/blockip2.txt" was not found.
04.02.15 09:09:57 BlockIP2: BlockExit QMgr="QSJHPT01" ChannelName="CHANNEL2" ConnectionName="127.0.0.1" Userid="root"
04.02.15 09:09:57 BlockIP2: BlockExit Connection refused, user don't match positive list xxx,yyy,zzz*,etc,mrmq,us???mq .
04.02.15 09:15:06 BlockIP2: BlockExit ConName "127.0.0.1" Pattern "10.1.*;10.1.11.*;etc;127.0.?.1" Flags "BlockMqmUsers=Y" users
"xxx,yyy,zzz*,etc,mrmq,root,us???mq".
04.02.15 09:15:06 BlockIP2: BlockExit Connection accepted QMgr="QSJHPT01" ChannelName="CHANNEL2" ConnectionName="127.0.0.1"
Userid="root"
```

Top of page

End User Agreement

You and your company are allowed to use and modify the program code as long that the code is not sold or build/included in a commercial product.

The intension of supplying the source code is to make it easier to WebSphere MQ professionals to keep their WMQ environments secure.

You are allowed to refer and/or put link on you web-site that points to http://www.mrmq.dk/BlockIP2. You are also allowed to include this reference in your written documentation, but we are allowed to move the website and links without any notice.

You are **NOT** allowed to place the download on your own internet-webserver for download. instead add a link to http://www.mrmq.dk/BlockIP2.

MRMQ.dk have no legal obligation if BlockIP2 causes your system any harm, BlockIP2 is written and is delivered on AS-IS basis, and we're trying to keep it working. When you install it on your own system, it's like a tool written by your self, and therefore on your own risk.

Top of page

Messages

In the z/OS version of the exit have we added the following message id's to help automation, and troubleshooting, and make documenation a bit easier

Message-id	Description

BLOCKIP-50I	Verification successful, connection accepted
BLOCKIP-51I	Verification successful, SSL have changes the userid, connection accepted
BLOCKIP-01E	Errors in SCYDATA specification. The specification don't match the required format. Check that SCYDATA() starts with 'FN=' or a valid pattern.
BLOCKIP-02E	Errors found in parameter file, check the extended error description
BLOCKIP-03E	Connection refused, User was found in the negative list.
BLOCKIP-04E	Connection refused, User was not found in positive list.
BLOCKIP-05E	Connection refused, ConName and User was not accepted in CON= specification
BLOCKIP-06E	Connection refused, The SSLPeer requested by CheckSSLList() was not successful
BLOCKIP-07E	Connection refused, for pattern [] user []
BLOCKIP-08E	Connection refused, SSLPeerName too long max [] was []
BLOCKIP-09E	Connection refused, CN requested blocked
BLOCKIP-10E	Connection refused for blank user identifier
BLOCKIP-11E	Connection refused for system user identifier (mqm)
BLOCKIP-12E	Connection refused for system user identifier (MUSR_MQMADM)
BLOCKIP-13E	Connection refused for system user identifier(musr_mqmadm)
BLOCKIP-14E	Connection refused, Pattern string is too long, max[] was []
BLOCKIP-15E	Connection refused, Users string is too long, max[], was []
BLOCKIP-16E	Connection refused, CON Spec string is too long, max [], was []
BLOCKIP-17E	Connection refused, SSL Spec string is too long, max[], was []
BLOCKIP-18E	Connection refused, BlockUsers string is too long, max[], was []

BLOCKIP-19E	Error - Unknown Tag []
BLOCKIP-20E	Connection refused, invalid file name pattern specified: FN=<filename...>; []
BLOCKIP-21E	Connection refused, Specified file [] was not found
BLOCKIP-22E	Connection refused, Users string is too long, max[], was []
BLOCKIP-30E	Connection refused, Required ClientExit missing
BLOCKIP-31E	Connection refused, Wrong credentials supplied.
BLOCKIP-32E	Connection refused, Maximum number of channels was exceeded.

The following are trademarks of International Business Machines Corporation:
IBM, MERVA, MQSeries, WebSphere, WBIFN, Object REXX, AIX, z/OS

Last updated: 2006.03.01.