[B] lob
[I] ntrusion
[M] onitor

Blob Integrity Monitor

Written By: [Name Removed]

Contents

About BIM:

Blob Integrity Monitor (BIM) is a client side tool for monitoring blob storage. For any blobs which change in blob storage, BIM will be able to detect and alert upon. This includes any introduction of new blobs and the removal of blobs as well as any changes in data. *Further functionality for monitoring unauthorized access to blob storage is in the development roadmap of the tool*. BIM has been designed to have a small foot print on system resources, to this goal many design choices were made. Additionally BIM does not persist sensitive information in memory, information such as blob storage keys or other information are only in memory during the time of the actual check and then disposed of after.

Copyright MIT

Getting Started:

Starting BlobIM - Running BlobIM.exe will automatically start BIM in graphical user interface (GUI) mode. BIM can also be started in silent mode, which essentially runs the tool from command line without the GUI. Note there is a manual shut off setting in “settings.config” called “ShutDownBIM” that can be used if silent mode is active. Once the tool is started it starts to monitor the blob storage indicated in configuration. BIM operates using the concept of “snapshots”, snapshots are essentially configuration files housing the details of blobs in blob storage in a metadata format.

Configuring BlobIM – BIM relies on two main configuration files “Settings.Config” and “Settings.Worker.Config” which both reside in the same directory as BlobIM.exe.

Snapshots – BIM uses the concept of snapshots. These act as a recording of the state of everything in a storage account at a particular point of time. By takes snapshots at different points of time then comparing them, BIM can then tell if a change occurs. The baseline snapshot in particular doubles as a configuration file as well to indicate if particular items on blob storage should or should not be monitored.

Functionality:
File Integrity Monitoring – This can be run real time or automatically simply by starting BIM which will enable and start FIM. FIM can be configured from the config files.

Permission Checking – BIM has the ability to review the access permissions of all blobs to detect if any blobs can be publically accessible or not. When run results are save to the same directory BIM is running in.

Storage Viewing – Per Snapshot/State - BIM can show a friendly view of snapshots for user convenience such as in HTML or CSV format.

Unauthorized Access Detection – Functionality not yet implemented.

Antivirus Scanning – As BIM downloads files to calculate the signature of each, Antivirus on the local machine can scan the same blob files.

Alerting&Logging – BIM has various logging and reporting functionalities all configurable through the config files:

- Detection Email Alerting – Sending email alerts on events detected
- Detection Email Report – On detection of events, send a CSV report by email
- Detection Logging Locally – if an event is detect, log the event to a local log
- Detection Logging to Windows Event Log – if an event is detected, log to windows event log
- Error Email Alerting – Sending email alerts if an error in BIM occurs
- Error Logging Locally – If an error in BIM occurs, log to local error log
- Error Logging to Windows Event Log – If an error in BIM occurs, log to windows event log
- Backup Email Alerting – Email a backup copy of a snapshot when an event is detected
- Backup To Local Copy – Save a backup copy of a snapshot locally when an event is detected

Heartbeat – BIM is able to send an email alert periodically announcing its uptime. The idea behind this is if a heartbeat email is missed, the tool is offline.

Warnings/Caution:

Large Blobs – BIM has two ways of analyzing blobs, download them first to analyze them or the second method is to simply use the metadata of the blob. In the case that the metadata of the blob becomes corrupt or altered, BIM first tries to do the download and analyze path first. The default size of files for which this happens is files under 3mb, the size can be change in the configuration file. If the size is too large, BIM will be spending a large time downloading files, additionally there are blob storage transaction costs to consider.

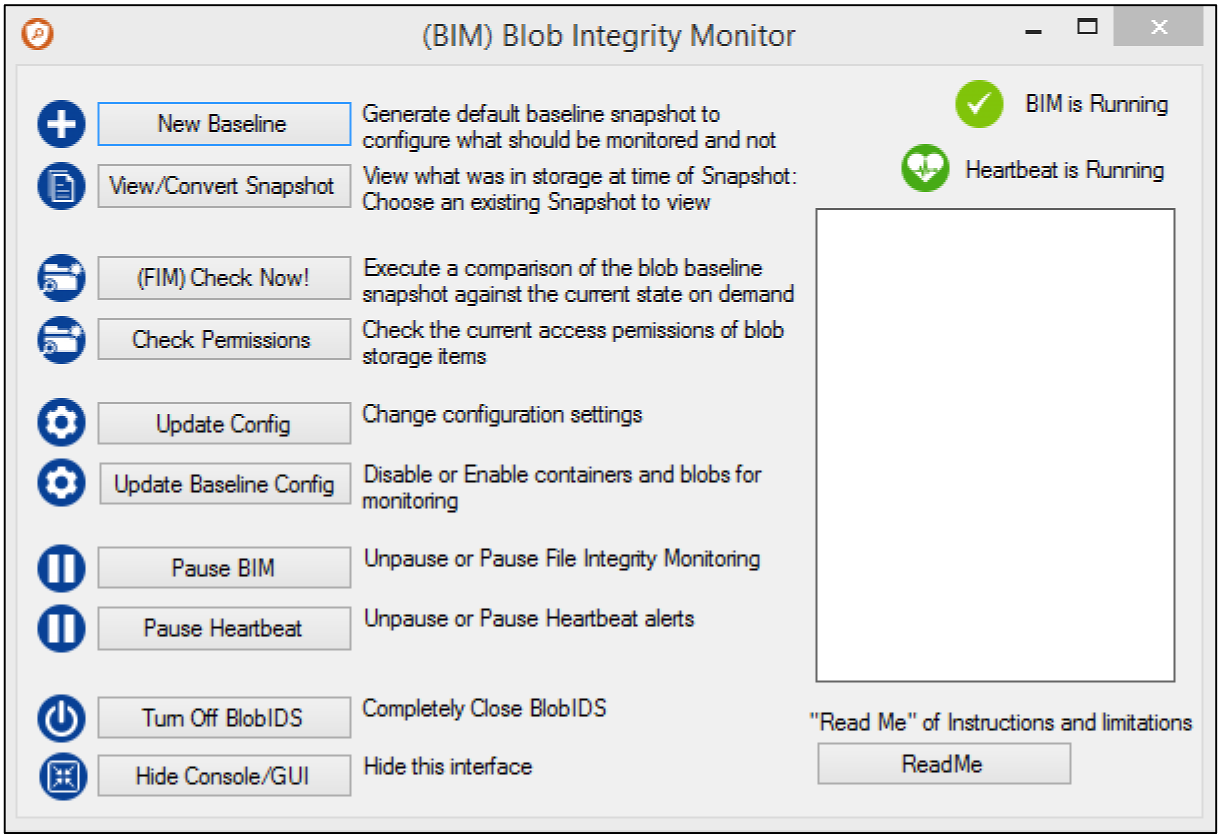
High Performance Scanning – For large files, reducing the download size to zero will cause BIM to use the metadata to determine if a blob has changed, this provides **a large increase in scanning speed.** While it does give insight if something has changed, it will not alert on what the change is. The primary detection will be around the date modified property of the blob. This could reflect a possible be a change in the metadata itself or it could be a change in the blob data, specification of which it is from this point is not possible. For smaller files, a download first to analyze the file can guarantee that a change in the same file can be accurately detected.

Important How To:

Snapshot Configuration – BIM uses “Snapshots” to record the state of blob storage accounts. These configuration files record details of blobs in blob storage at the time the Snapshot is taken. The Snapshot can be (optionally) configured such as “DoNotOverwrite="False"” which ensure that a snapshot will never be overwritten as well as options to disable monitoring on entire containers or specific blobs.
More details: Configure Snapshots

Integrity Check Against a Baseline – On the very first Integrity Check, BIM will use the baseline snapshot to compare against the current state of the storage account. If it is desired to know the difference between the baseline and the current state at any point in time, go to the working directory of BlobIM.exe and delete “BIM_Previous.Snapshot” before running the “(FIM) Check Now!” on demand comparison scan.

Graphical User Interface (GUI)



Generate default baseline snapshot to configure what should be monitored and not

View what was in storage at time of Snapshot: Choose an existing Snapshot to view

Execute a comparison of the blob baseline snapshot against the current state on demand

Check the current access pemissions of blob storage items

Change configuration settings

Disable or Enable containers and blobs for monitoring

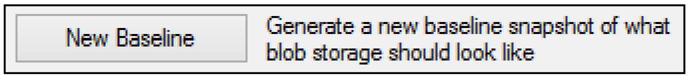
Unpause or Pause File Integrity Monitoring

Unpause or Pause Heartbeat alerts

Completely Close BlobIDS

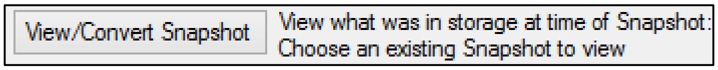
Hide this interface

"Read Me" of Instructions and limitations



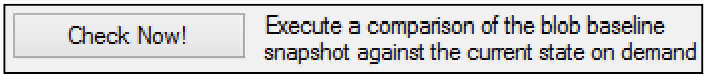
Generate a new baseline snapshot of what blob storage should look like

This is an on demand option to generate a new baseline snapshot. This snapshot will be used as a reference for what future states of blob storage should then be compared against to detect changes.



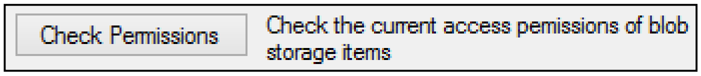
View what was in storage at time of Snapshot: Choose an existing Snapshot to view

This will allow viewing snapshot files in various formats such as HTML or CSV. Only the original format of ".Snapshot" can be consumed by BIM, the additional formats are for user convenience.



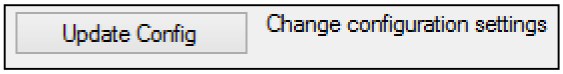
Execute a comparison of the blob baseline snapshot against the current state on demand

This is an on demand option to check blob storage for any changes. A comparison will be made against the latest baseline snapshot to detect if any blobs have changed, been introduced, or removed.



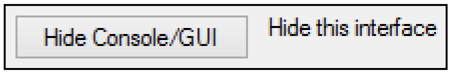
Check the current access pemissions of blob storage items

This is an on demand option to check the access permissions of all blobs in blob storage. It will detect of blobs can be accessed anonymously and generate a report to the running directory of BIM as "BlobPermissionsReport.csv"


Update Config

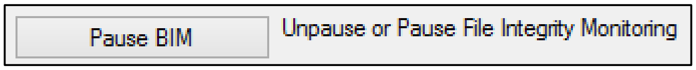
Change configuration settings

This is an on demand option to edit the configuration files. Items in red must be modified/filled out, however all fields should be reviewed for correct information.


Hide Console/GUI

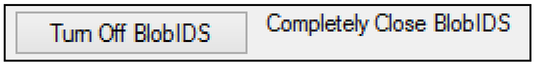
Hide this interface

This will hide the Graphical User Interface and run the tool in the background. The GUI can be displayed again from the tray icon menu by right clicking on the BIM icon.


Pause BIM

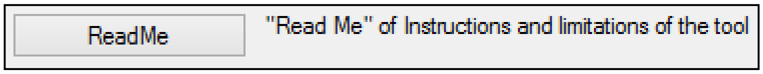
Unpause or Pause File Integrity Monitoring

This will allow File Integrity Monitoring to be Temporarily Disabled, FIM is automatically started when BIM is run.


Turn Off BlobIDS

Completely Close BlobIDS

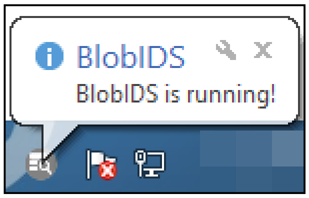
Completely shuts down BIM.


ReadMe

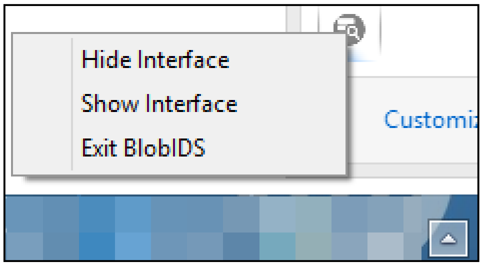
"Read Me" of Instructions and limitations of the tool

Opens this ReadMe.

GUI Tray Icon/Menu


BlobIDS

BlobIDS is running!



BIM can also be access from the system tray. Different notifications can additionally be displayed from the tray such as when an event is detected or the occurrence of an error. Enabling or Disabling of tray notification can be configured in the config files.

Hide Interface

This will hide the Graphical User Interface and run the tool in the background. The GUI can be displayed again from the tray icon menu by right clicking on the BIM icon.

Show Interface

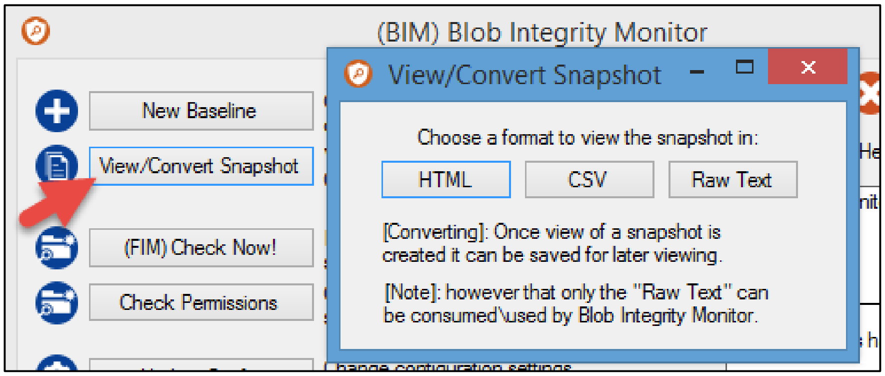
This will display the Graphical User Interface and when the tool is running in the background.

Exit BlobIDS

Completely shuts down BIM.

View/Convert Snapshots

Snapshots can be viewed in various formats such as HTML and CSV. Access to this functionality is through the GUI by clicking on the "View/Convert Snapshot" button and choosing a format.

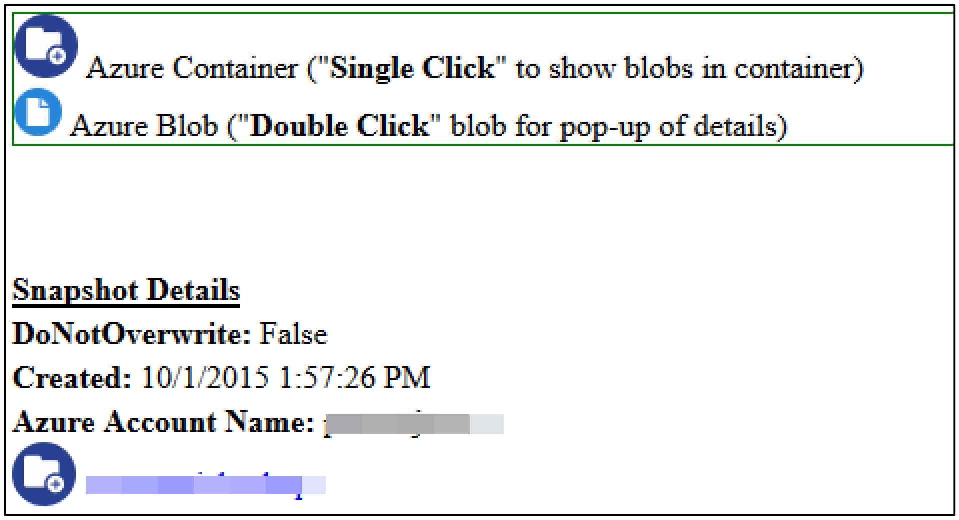


Choose a format to view the snapshot in:

[Converting]: Once view of a snapshot is created it can be saved for later viewing.

[Note]: however that only the "Raw Text" can be consumed\used by Blob Integrity Monitor.

E.g: If HTML is chosen it will produce a view of the snapshot that was recorded.



Azure Container ("**Single Click**" to show blobs in container)

Azure Blob ("**Double Click**" blob for pop-up of details)

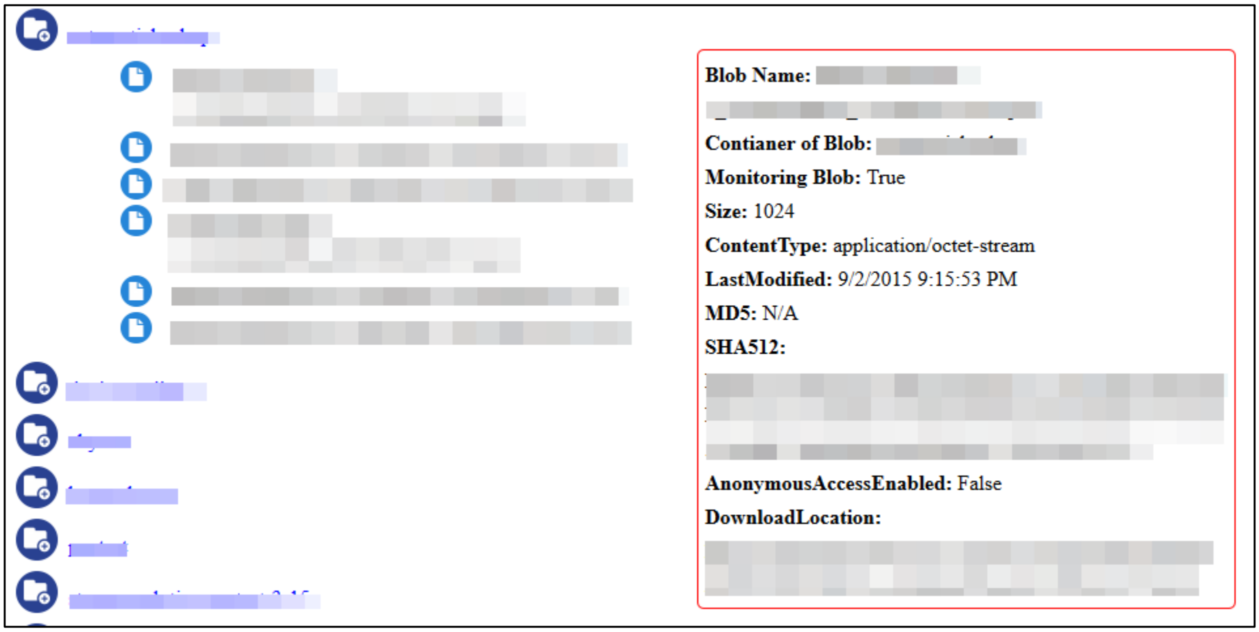
<u>Snapshot Details</u>

DoNotOverwrite: False

Created: 10/1/2015 1:57:26 PM

Azure Account Name: [redacted]

[redacted]



[redacted]

Blob Name: [redacted]

Contianer of Blob: [redacted]

Monitoring Blob: True

Size: 1024

ContentType: application/octet-stream

LastModified: 9/2/2015 9:15:53 PM

MD5: N/A

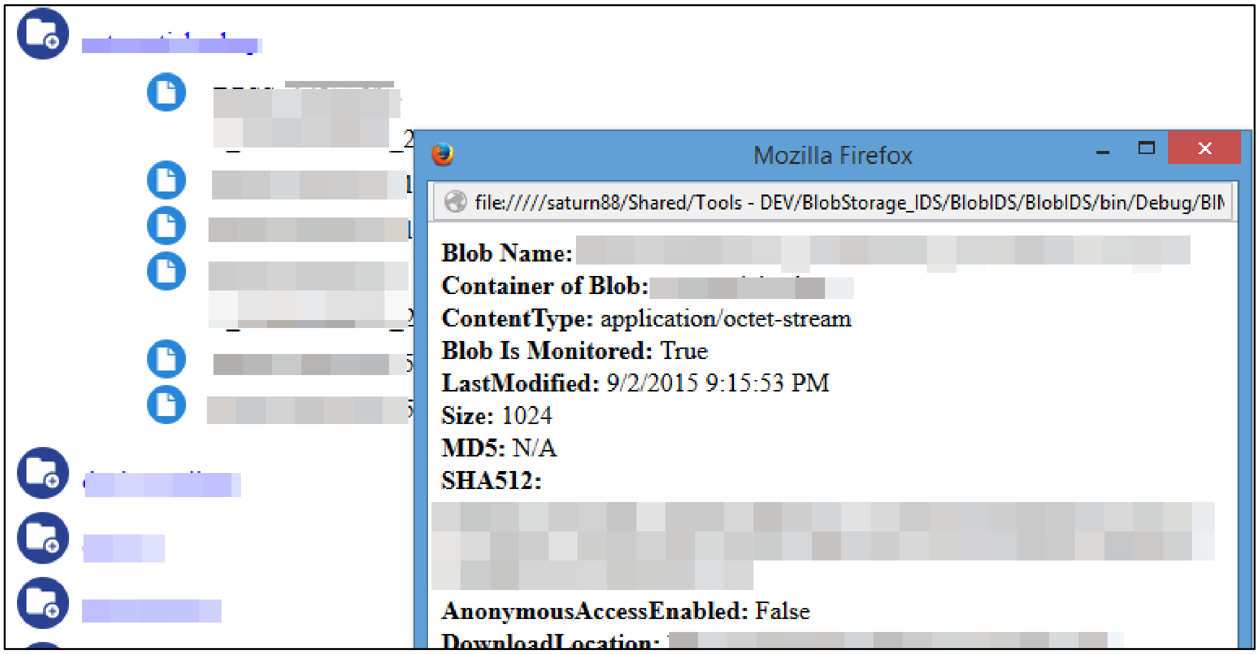
SHA512:

[redacted]

AnonymousAccessEnabled: False

DownloadLocation:

[redacted]



[redacted]

file://///saturn88/Shared/Tools - DEV/BlobStorage_IDS/BlobIDS/BlobIDS/bin/Debug/BIN

Blob Name: [redacted]

Container of Blob: [redacted]

ContentType: application/octet-stream

Blob Is Monitored: True

LastModified: 9/2/2015 9:15:53 PM

Size: 1024

MD5: N/A

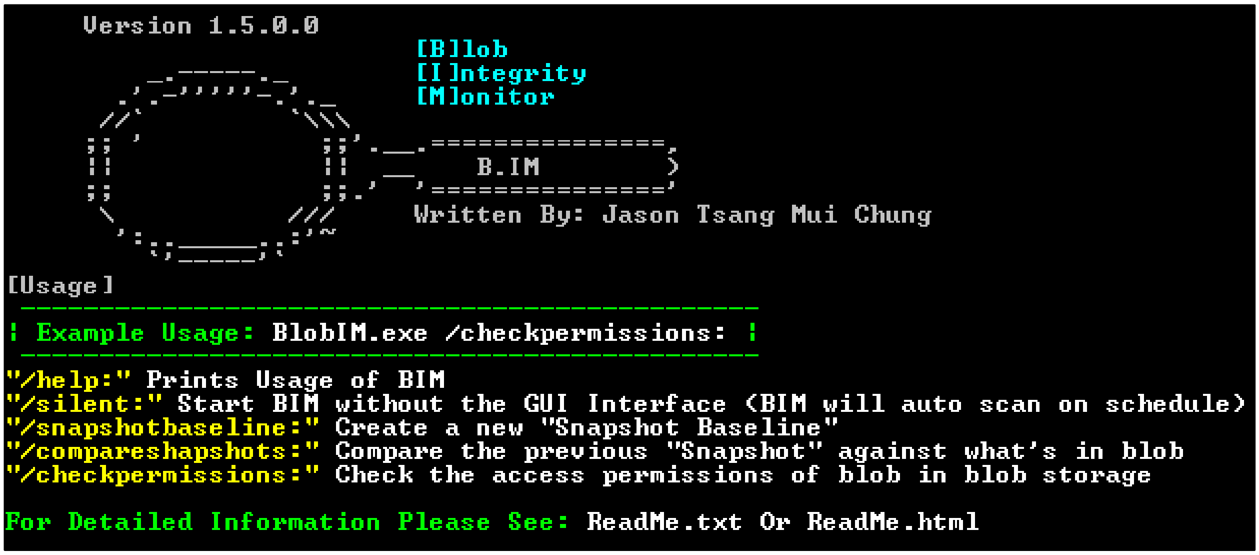
SHA512:

[redacted]

AnonymousAccessEnabled: False

DownloadLocation: [redacted]

Command Line Usage



```
     Version 1.5.0.0
                          [B]lob
         _.------._       [I]ntegrity
       ,'_,,,,,,_.',      [M]onitor
     .'.          '._
    //'             '\\\
   ;;  '            ;;'.__.===============,
   ||               ||  __      B.IM      >
   ;;               ;;.' __'==============' 
    \              ///    Written By: Jason Tsang Mui Chung
     ':          .'~
       ';;______;:'
[Usage]
 ------------------------------------------------
| Example Usage: BlobIM.exe /checkpermissions: |
 ------------------------------------------------
"/help:" Prints Usage of BIM
"/silent:" Start BIM without the GUI Interface (BIM will auto scan on schedule)
"/snapshotbaseline:" Create a new "Snapshot Baseline"
"/compareshapshots:" Compare the previous "Snapshot" against what's in blob
"/checkpermissions:" Check the access permissions of blob in blob storage

For Detailed Information Please See: ReadMe.txt Or ReadMe.html
```

Generally Command Line usage allows for two use case scenarios:

1) On Demand – The basic functionalities of BIM can be run on demand such as taking snapshots, comparing snapshots, or checking blob permissions.
2) Silent – Bid can be run without the GUI starting up, it will run automatically using what is set in the configuration files.

`"/help:" Prints Usage of B.IDS`

Prints the Usage page of the various parameters/flags that can be used.

`"/silent:" Start B.IDS without the GUI Interface`

Run BIM without the GUI, automatically runs detection based on scheduling using configuration from config files.

`"/snapshotbaseline:" Create a new "Snapshot Baseline"`

Create a new baseline snapshot of blob storage, this is a point in time from which future states of blob storage will be compared against.

`"/compareshapshots:" Compare "Snapshot Baseline" against what's in blob`

This will compare the baseline snapshot against the state of blob storage when BIM is run.

`"/checkpermissions:" Check the access permissions of blob in blob storage`

This will generate a report of all blobs in blob storage which might have anonymous access enabled.

Config Files

Settings.Worker.Config

E.g: `<setting name="Setting Name">Setting Value that can be changed</setting>`

`<setting name="BlobStorage_Account_Name">[Required To Be Changed]</setting>`	BlobStorage Account to be Monitored

<setting name="BlobStorage_Account_Key">[Required To Be Changed]</setting>	Key for the BlobStorage Account to be monitored, if DPAI Encrypted must be base64 encoded and make sure BlobStorage_Account_Key_Encrypted is set to true, if NOT Encrypted make sure BlobStorage_Account_Key_Encrypted is set to false
<setting name="BlobStorage_Account_Key_Encrypted">True</setting>	If Account Key in this setting file is Encrypted with DPAPI, if set to False then BIM will try to use the key as "clear text" (without decrypting it)
<setting name="BlobStorage_Account_Environment">blob.core.windows.net</setting>	Set the Blob Storage Endpoint environment to connect to.
<setting name="Email_SMTPServer_Generic">smtp-mail.outlook.com</setting>	The SMTP email server to send out email alerts
<setting name="Email_SMTPServer_Port">587</setting>	Port of SMTP Email Server
<setting name="Email_Account">[Required To Be Changed]</setting>	Email Account to login to SMTP server
<setting name="Email_Password">[Required To Be Changed]</setting>	Password for Email Account to login to SMTP Server, if DPAI Encrypted must be base64 encoded and make sure Email_Password_Encrypted is set to true, if NOT Encrypted make sure Email_Password_Encrypted is set to false
<setting name="Email_Password_Encrypted">True</setting>	If Email Password in this setting file is Encrypted with DPAPI, if set to False then BIM will try to use the password as "clear text" (without decrypting it)
<setting name="Email_SMTPServer_WindowsCredentials">smtp-mail.outlook.com</setting>	[Not Implemented] If Windows authentication is enabled, SMTP server to use with windows based Authentication
<setting name="Email_SendUsing_Generic">True</setting>	Enable or Disable sending email with SMTP and regular authentication
<setting name="Email_SendUsing_WindowsCredentials">False</setting	[Not Implemented] Enable or Disable sending email with SMTP using windows based authentication
<setting name="Email_Sender">[Required To Be Changed]</setting>	The sender of who email alerts will appear they are from

<setting name="Email_Recipient">[Required To Be Changed]</setting	Who to send emails to. To add multiple separate each email with ';'
<setting name="Email_Subject_Alerts">BlobIM Event Detected</setting>	Specify the subject of email Detection based Alerts
<setting name="Email_Subject_HeartBeat">BlobIM HeartBeat</setting>	Specify the subject of email heart beat alerts
<setting name="Email_Subject_Errors">BlobIM Error Occurred</setting>	Specify the subject of email Error based alerts
<setting name="Email_Subject_Reporting">BlobIM Report</setting>	Specify the subject of email Report based alerts
<setting name="Email_Subject_Backups">BlobIM Backup</setting>	Specify the subject of email Backup file based alerts

Settings.Config

E.g: <setting name="**Setting Name**">Setting Value that can be changed</setting>

<setting name="ShutDownBIM">False</setting>	This is a manual override setting for turning off BIM. When set to true, on the next scanning interval the BIM engine will exit instead.
<setting name="DownloadPath">./</setting>	Path of where to allow the tool to store temp files
<setting name="MaxDownloadSize">3000000</setting>	Maximum size a file can be to be downloaded for detection purposes, 3000000 = 3mb (keep in mind transaction costs of download files)
<setting name="SnapShotFileName_Baseline">./BIM_Baseline.Snapshot</setting>	The state of blob storage baseline is saved as

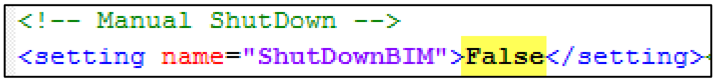
	"snapshots", specify the file name for that here
<setting name="SnapShotFileName_Current">./BIM_Current.Snapshot</setting>	The state of blob storage during every check is saved as "snapshots", specify the file name for that here
<setting name="SnapShotFileName_FileAccess">./BIM_FileAccess.Snapshot</setting>	The state of blob storage identifing file permissions saved as "snapshot", specify the file name for that here
<setting name="DetectionPeriod_EveryNumberof_Hours">1</setting>	(Hours and minutes can be combine for the same settings) specify periodically how many hours to do a check at, zero will effectively disable this setting
<setting name="DetectionPeriod_EveryNumberof_Minutes">0</setting>	(Hours and minutes can be combine for the same settings) specify periodically how many minutes to do a check at, zero will effectively disable this setting
<setting name="EventLog_FileName">./Report.log</setting>	FileName of the log of

	Detected Events
<setting name="EventLog_SaveEvents">True</setting>	Enable or Disable saving detected events to log
<setting name="EventLog_MaxSize">100</setting>	The Maximum size in MB a log file can be before it is rolled over
<setting name="EventLog_WriteTo_WindowsApplicationLog">False</setting>	Enable or Disable recording Detected Events in the Windows Application Log
<setting name="EventLog_ShowTaskBarMessage">True</setting>	Enable or Disable taskbar notifications
<setting name="Report_FileName_html">./Report.html</setting>	File name of the Detected Events Report in HTML format, this is only a temporary file which is deleted after sending
<setting name="SendReportAs_HTML">True</setting>	Enable or Disable sending reports in HTML format
<setting name="Report_FileName_csv">./Report.csv</setting>	File name of the Detected Events Report in CSV format, this is only a temporary file which is deleted after sending
<setting name="SendReportAs_CSV">True</setting>	Enable or Disable sending

	reports in HTML format
<setting name="SaveBackupOfSnapshotsOnEventFound">True</setting>	Enable or Disable saving the "snapshot" as a backup if an Event is Detected
<setting name="EmailBackupOfSnapshotsOnEventFound">True</setting>	Enable or Disable Emailing a copy of the "snapshot" if an Event is Detected
<setting name="ErrorLog_Path">./Error.log</setting>	File name and path of were to save error log files
<setting name="ErrorLog_WriteTo_LocalLog">True</setting>	Enable or Disable weather errors should be saved to log file
<setting name="ErrorLog_WriteTo_WindowsApplicationLog">True</setting>	Enable or Disable if errors should be written to windows Application Log
<setting name="ErrorLog_SendEmail_PerError">True</setting>	Enable or Disable if emails should be sent when a error occurs
<setting name="ErrorLog_ShowTaskBarMessage">True</setting>	Enable or Disable taskbar notifications for errors
<setting name="HealthMonitoring_SendEmail_HeartBeatPeriodicallyByHour">1</setting>	(Hours and minutes can be combine for the same settings) specify periodically how many hours to send a

	health status email at, zero will effectively disable this setting
`<setting name="HealthMonitoring_SendEmail_HeartBeatPeriodicallyByMinutes">0</setting>`	(Hours and minutes can be combine for the same settings) specify periodically how many minutes to send a health status email at, zero will effectively disable this setting

Manual ShutDown



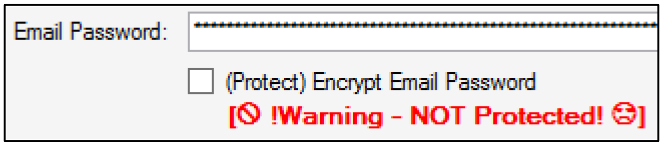
```
<!-- Manual ShutDown -->
<setting name="ShutDownBIM">False</setting>
```

This is a manual override setting for turning off BIM. When set to true, on the next scanning interval the BIM engine will exit instead. Such a setting is useful when using BIM in silent mode, if the input for the process can't be found to issue a quit or exit, instead of having to kill the process the engine can be shut down from the configuration file.

(Protection) Encryption of Config Secrets with DPAI

BIM uses DPAI to protect secrets which are stored in its configuration files. This (protection) encryption will ensure that if the configuration file were disclosed, anyone who were to obtain or steal the configuration should not directly be able to gain the blob storage account key or email account password in the configuration file.

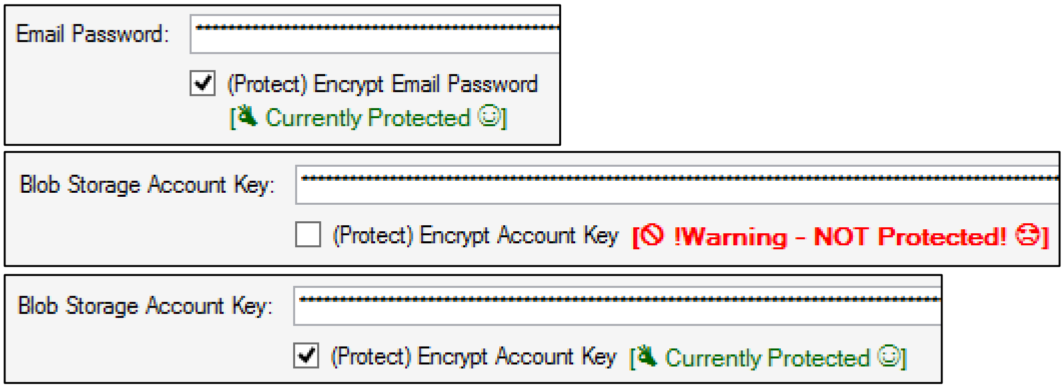
- Using the settings interface in BIM will alert the user when a secret is unprotected or if it is already protected in the configuration file.



Email Password: **

□ (Protect) Encrypt Email Password

[🛇 !Warning - NOT Protected! ☹]



Email Password: **

☑ (Protect) Encrypt Email Password
[Currently Protected ☺]

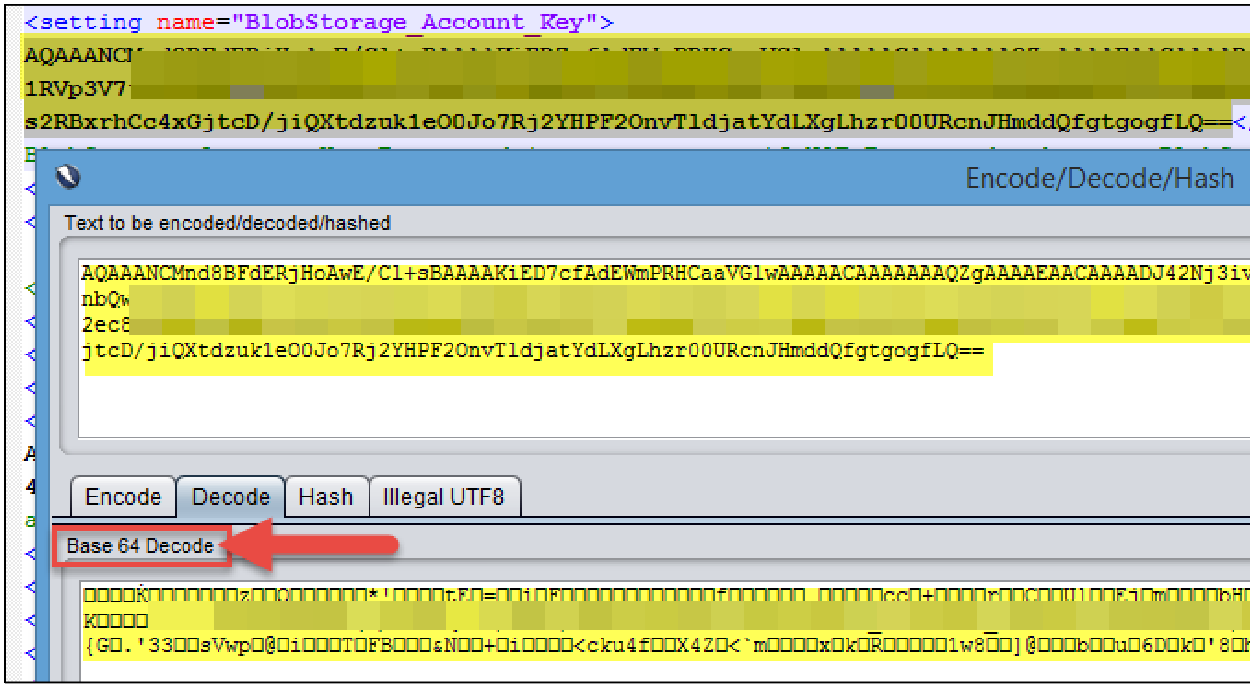
Blob Storage Account Key: **

☐ (Protect) Encrypt Account Key [⊘ !Warning - NOT Protected! ☹]

Blob Storage Account Key: **

☑ (Protect) Encrypt Account Key [Currently Protected ☺]

- (protected) encrypted secret in configuration files are base64 encoded after they are encrypted:



```
<setting name="BlobStorage_Account_Key">
AQAAANCI[redacted]
1RVp3V7
s2RBxrhCc4xGjtcD/jiQXtdzuk1eO0Jo7Rj2YHPF2OnvTldjatYdLXgLhzr00URcnJHmddQfgtgogfLQ==<
```

Text to be encoded/decoded/hashed

```
AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAKiED7cfAdEWmPRHCaaVGlwAAAAACAAAAAAAQZgAAAAEAACAAAADJ42Nj3iv
nbQw[redacted]
2ec8
jtcD/jiQXtdzuk1eO0Jo7Rj2YHPF2OnvTldjatYdLXgLhzr00URcnJHmddQfgtgogfLQ==
```

```
□□□□K□□□□□□□□z□□O□□□□□□□*'□□□□tF□=□□i□F□□□□□□□□□□□□□f□□□□□□ □□□□□cc□+□□□□r□□C□□U□□□Fj□m□□□□bH
K□□□□[redacted]
{G□.'33□□sVwp□@□i□□□T□FB□□□&N□□+□i□□□□<cku4f□□X4Z□<`m□□□□x□k□R□□□□□1w8□□]@□□□b□□u□6D□k□'8□
```

Warning – User Account Dependency with (Protection) Encryption of Config Secrets

Note that the user account which BIM was run under when enabling protection/encryption in the config file will have to be the user account that BIM runs under in the future. That user account will be the only account which will allow BIM to understand the secrets in the config file afterwards.

Manually Creating (Protected) Encrypted Secrets for Config File with DPAI

It is possible to manually encrypted secrets using DPAI and then manually entering it into the configuration file instead of using the GUI settings interface. This may be necessary in certain scenarios such as setting up the tool on an operating system with no graphical interface. To do so follow the steps below:

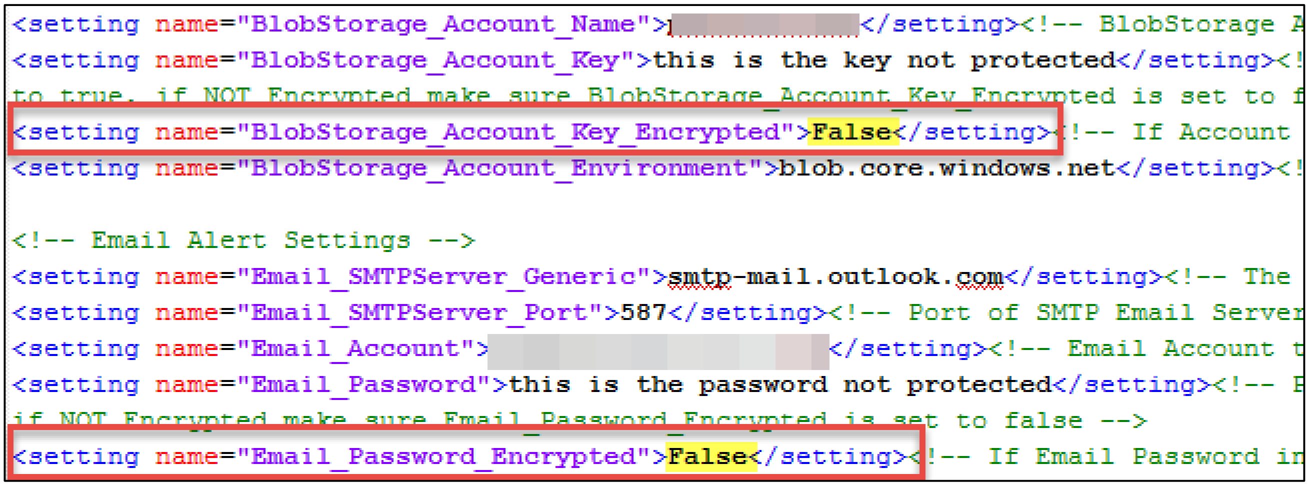
1. Manually encrypt secrets using DPAPI https://msdn.microsoft.com/en-us/library/2fh8203k%28v=vs.110%29.aspx, additional background info
2. Base64 encode the encrypted secrets https://msdn.microsoft.com/en-us/library/dhx0d524%28v=vs.110%29.aspx
3. Insert the base64 encoded encrypted secret into the config where the respective secret should be ("BlobStorage_Account_Key" & "Email_Password") in settings.worker.config
4. Make sure that "BlobStorage_Account_Key_Encrypted" and "Email_Password_Encrypted" are both set to “True”
5. Make sure that BIM is running using the user account which was used during the DPAI encryption process in step “1”

Use Plain Text Secrets - Disable Encryption of Secrets in Config File

If it is desired to disable protection/encryption of secrets in the configuration files it is possible to do so manually or through the BIM GUI settings interface.

Manually:

Make sure to enter the secrets in their original form (plain text and unprotected/unencrypted), then set the respective encrypted settings to false.

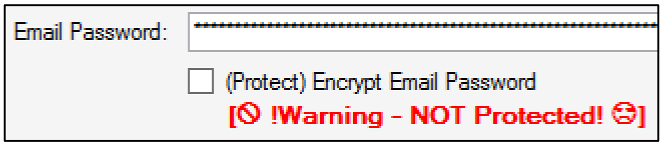


```
<setting name="BlobStorage_Account_Name">[redacted]</setting><!-- BlobStorage A
<setting name="BlobStorage_Account_Key">this is the key not protected</setting><!
to true. if NOT Encrypted make sure BlobStorage_Account_Key_Encrypted is set to f
<setting name="BlobStorage_Account_Key_Encrypted">False</setting><!-- If Account
<setting name="BlobStorage_Account_Environment">blob.core.windows.net</setting><!

<!-- Email Alert Settings -->
<setting name="Email_SMTPServer_Generic">smtp-mail.outlook.com</setting><!-- The
<setting name="Email_SMTPServer_Port">587</setting><!-- Port of SMTP Email Server
<setting name="Email_Account">[redacted]</setting><!-- Email Account t
<setting name="Email_Password">this is the password not protected</setting><!-- F
if NOT Encrypted make sure Email_Password_Encrypted is set to false -->
<setting name="Email_Password_Encrypted">False</setting><!-- If Email Password in
```

Through the GUI Settings Interface:

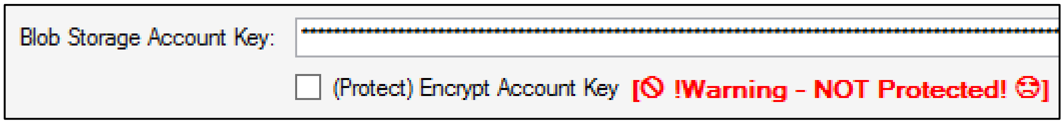
In the settings interface of the GUI simple enter the passwords as you normally would and ensure the protect/encrypt options and unchecked, then click save.



Email Password: **

□ (Protect) Encrypt Email Password

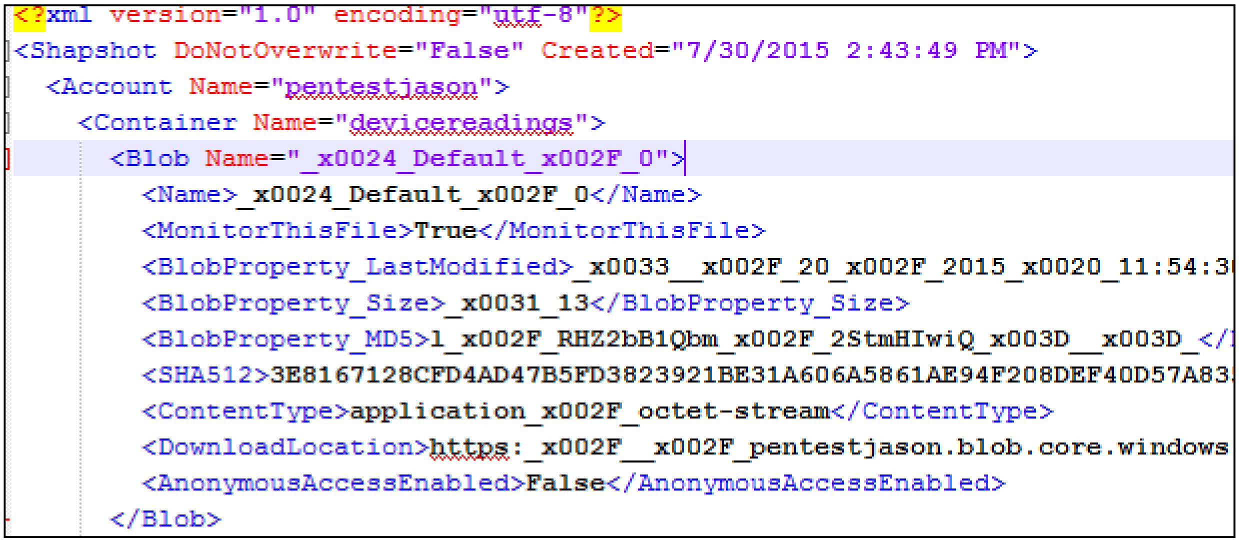
[🛇 !Warning - NOT Protected! ☹]



Blob Storage Account Key: **

□ (Protect) Encrypt Account Key [🛇 !Warning - NOT Protected! ☹]

Snapshots

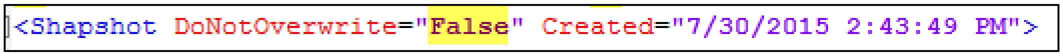


```
<?xml version="1.0" encoding="utf-8"?>
<Shapshot DoNotOverwrite="False" Created="7/30/2015 2:43:49 PM">
  <Account Name="pentestjason">
    <Container Name="devicereadings">
      <Blob Name="_x0024_Default_x002F_0">
        <Name>_x0024_Default_x002F_0</Name>
        <MonitorThisFile>True</MonitorThisFile>
        <BlobProperty_LastModified>_x0033__x002F_20_x002F_2015_x0020_11:54:3
        <BlobProperty_Size>_x0031_13</BlobProperty_Size>
        <BlobProperty_MD5>1_x002F_RHZ2bB1Qbm_x002F_2StmHIwiQ_x003D__x003D_</
        <SHA512>3E8167128CFD4AD47B5FD3823921BE31A606A5861AE94F208DEF40D57A83
        <ContentType>application_x002F_octet-stream</ContentType>
        <DownloadLocation>https:_x002F__x002F_pentestjason.blob.core.windows
        <AnonymousAccessEnabled>False</AnonymousAccessEnabled>
      </Blob>
```

The state of a blob storage is recorded in a snapshot file. There are three places that a snapshot file can be modified and configured:

Config File Level:

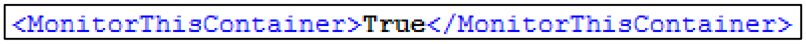
"**DoNotOverwrite=**" can be set to ensure that a snapshot will never be overwritten. This is available to help ensure mistakes will not happen.



```
<Shapshot DoNotOverwrite="False" Created="7/30/2015 2:43:49 PM">
```

Container Level Monitoring:

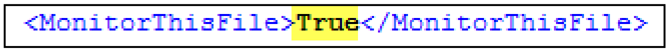
In a snapshot entire containers can be configured to not be monitored.



```
<MonitorThisContainer>True</MonitorThisContainer>
```

Blob Level Monitoring:

In a snapshot individual blobs can be configured to not be monitored.



```
<MonitorThisFile>True</MonitorThisFile>
```

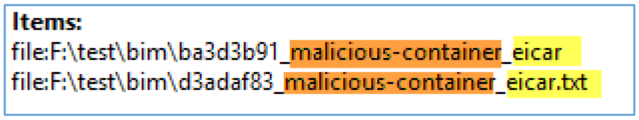
Antivirus Scanning

Client side antivirus scanning can be leveraged when using BIM. BIM will attempt to download various blobs to a designated folder to calculate a signature of the blob for comparisons in different points of time. Naturally as BIM downloads these files, if the designated folder for the files to be temporarily downloaded to, are in scope of the machines Antivirus scanning, the Antivirus can essentially scan blobs which are reflected in the storage account. The only limitations of this are the download size limit of blobs which is configurable.

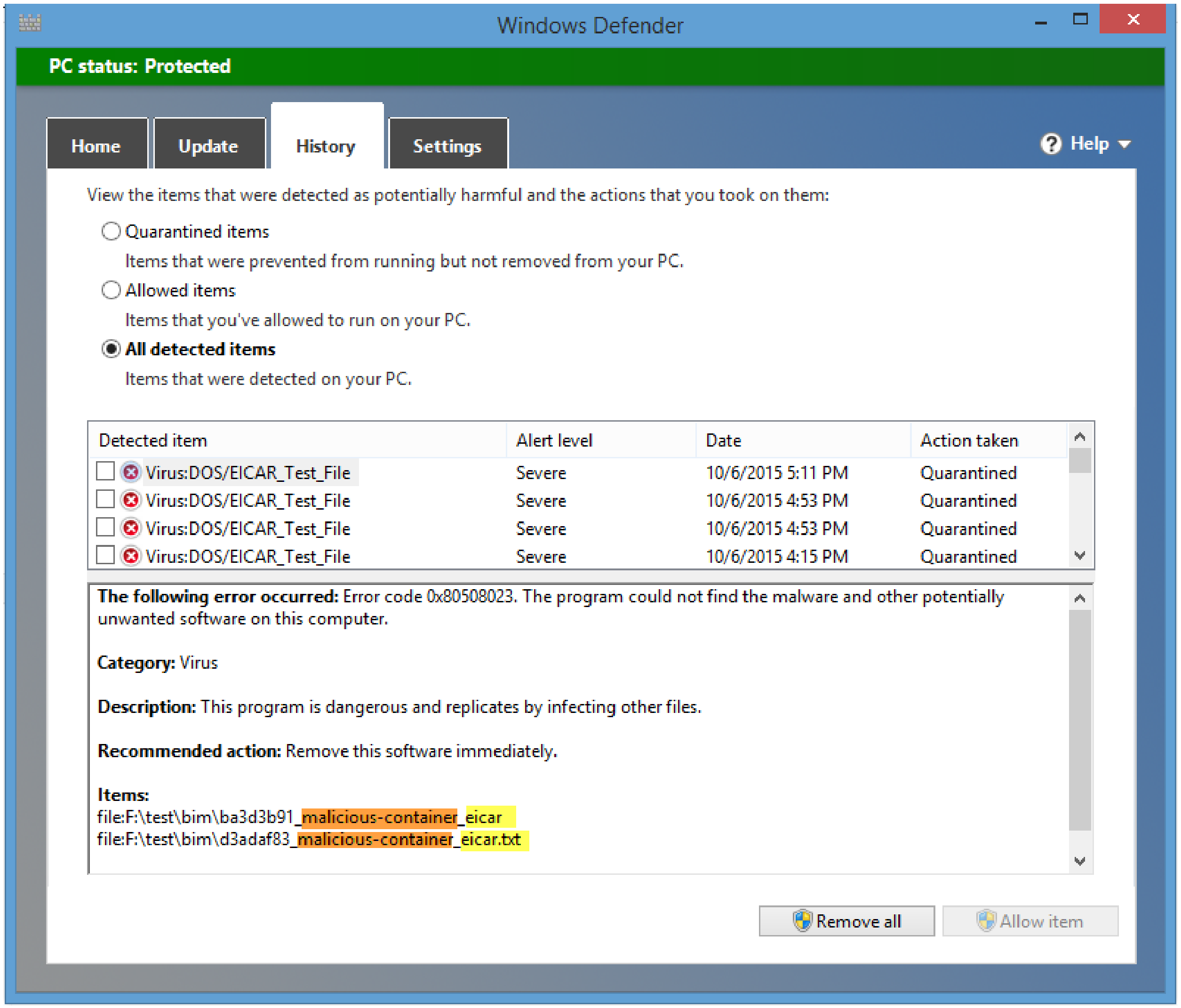
If Antivirus does detect a particular blob, the file name that the Antivirus will record will be of the following format:

Path\RandomString_ContainerName_BlobName

E.g:



Items:
file:F:\test\bim\ba3d3b91_malicious-container_eicar
file:F:\test\bim\d3adaf83_malicious-container_eicar.txt



View the items that were detected as potentially harmful and the actions that you took on them:

○ Quarantined items
Items that were prevented from running but not removed from your PC.

○ Allowed items
Items that you've allowed to run on your PC.

● **All detected items**
Items that were detected on your PC.

Detected item	Alert level	Date	Action taken
□ Virus:DOS/EICAR_Test_File	Severe	10/6/2015 5:11 PM	Quarantined
□ Virus:DOS/EICAR_Test_File	Severe	10/6/2015 4:53 PM	Quarantined
□ Virus:DOS/EICAR_Test_File	Severe	10/6/2015 4:53 PM	Quarantined
□ Virus:DOS/EICAR_Test_File	Severe	10/6/2015 4:15 PM	Quarantined

The following error occurred: Error code 0x80508023. The program could not find the malware and other potentially unwanted software on this computer.

Category: Virus

Description: This program is dangerous and replicates by infecting other files.

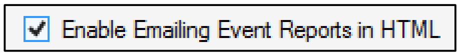
Recommended action: Remove this software immediately.

Items:
file:F:\test\bim\ba3d3b91_malicious-container_eicar
file:F:\test\bim\d3adaf83_malicious-container_eicar.txt

Alerting and Reporting

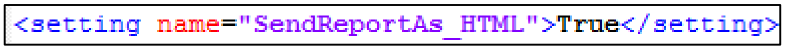
[Email] HTML Report

Configured from the GUI “Update Config” option under “Settings.Config” tab:



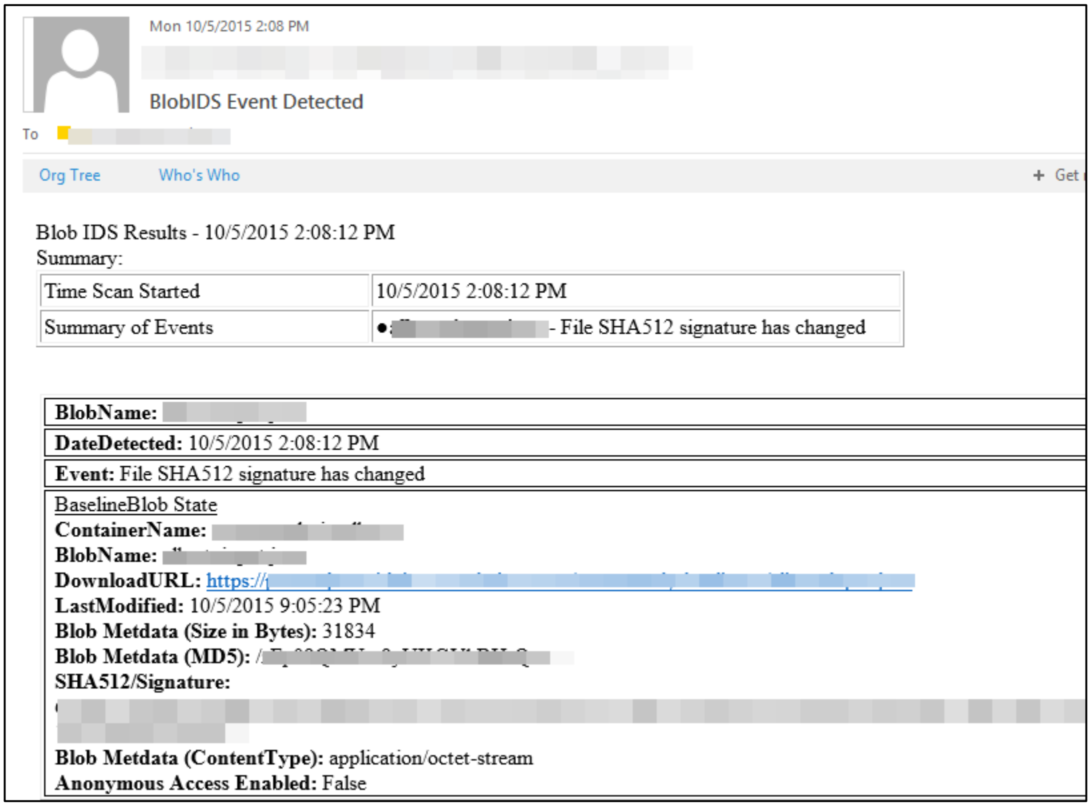
☑ Enable Emailing Event Reports in HTML

Or also updated directly in the Settings.Config file:



```
<setting name="SendReportAs_HTML">True</setting>
```

Email:



Mon 10/5/2015 2:08 PM

[redacted]

BlobIDS Event Detected

To [redacted]

Blob IDS Results - 10/5/2015 2:08:12 PM

Summary:

Time Scan Started	10/5/2015 2:08:12 PM
Summary of Events	● [redacted] - File SHA512 signature has changed

BlobName: [redacted]

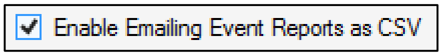
DateDetected: 10/5/2015 2:08:12 PM

Event: File SHA512 signature has changed

BaselineBlob State
ContainerName: [redacted]
BlobName: [redacted]
DownloadURL: https://[redacted]
LastModified: 10/5/2015 9:05:23 PM
Blob Metdata (Size in Bytes): 31834
Blob Metdata (MD5): /[redacted]
SHA512/Signature:
[redacted]
Blob Metdata (ContentType): application/octet-stream
Anonymous Access Enabled: False

[Email] CSV Report

Configured from the GUI “Update Config” option under “Settings.Config” tab:

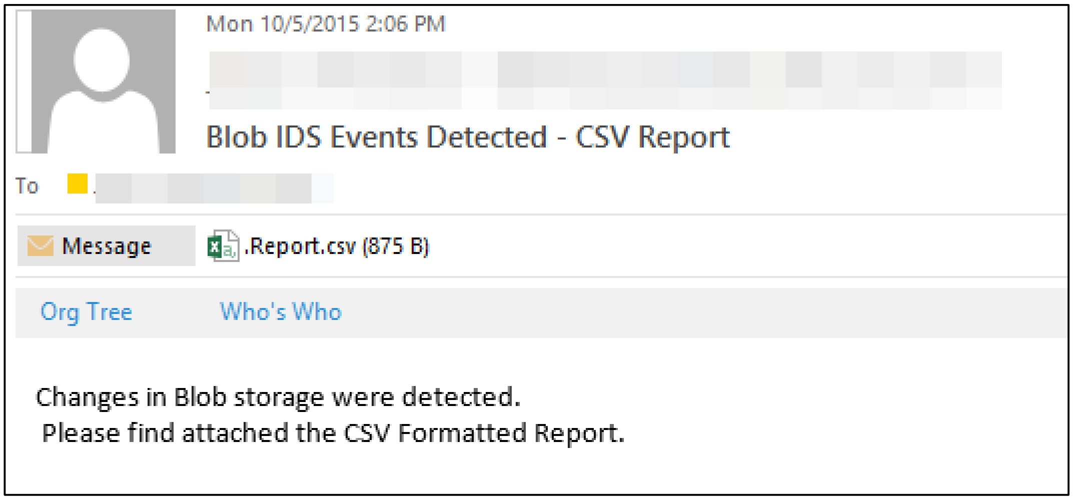


☑ Enable Emailing Event Reports as CSV

Or also updated directly in the Settings.Config file:

```
<setting name="SendReportAs_CSV">False</setting>
```

Email:



Mon 10/5/2015 2:06 PM

[redacted]

Blob IDS Events Detected - CSV Report

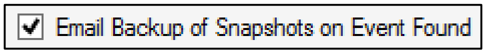
To [redacted]

.Report.csv (875 B)

Changes in Blob storage were detected.
Please find attached the CSV Formatted Report.

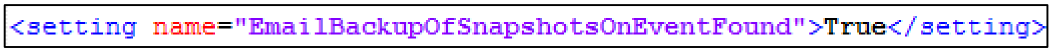
[Email] Backup Snapshot

Configured from the GUI “Update Config” option under “Settings.Config” tab:



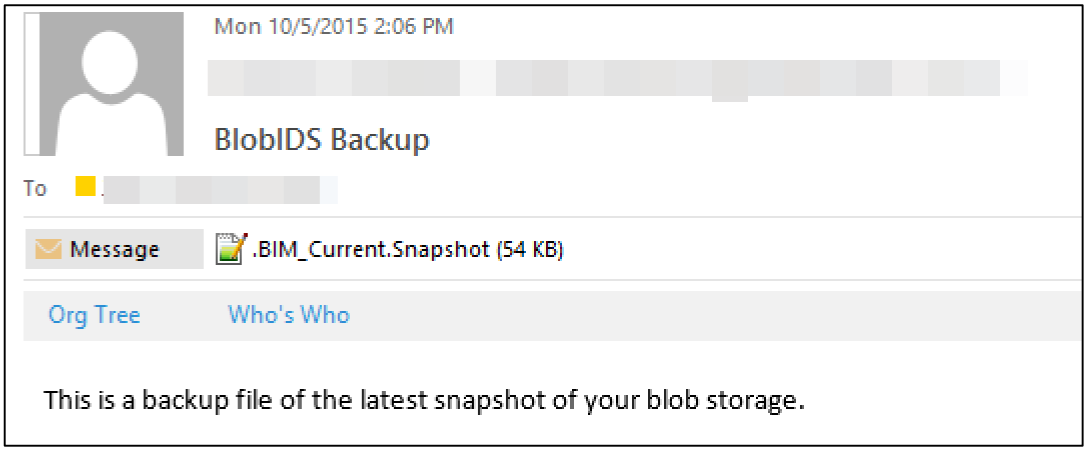
☑ Email Backup of Snapshots on Event Found

Or also updated directly in the Settings.Config file:



```
<setting name="EmailBackupOfSnapshotsOnEventFound">True</setting>
```

Email:



Mon 10/5/2015 2:06 PM

[redacted]

BlobIDS Backup

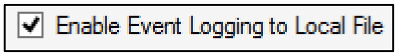
To [redacted]

.BIM_Current.Snapshot (54 KB)

This is a backup file of the latest snapshot of your blob storage.

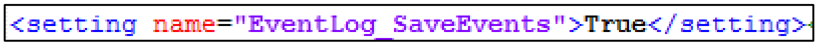
[Local] Log File

Configured from the GUI “Update Config” option under “Settings.Config” tab:



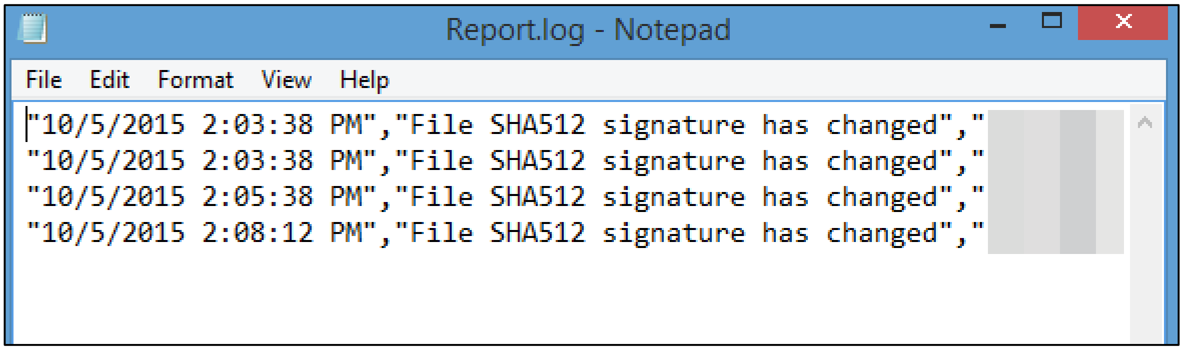
☑ Enable Event Logging to Local File

Or also updated directly in the Settings.Config file:



```
<setting name="EventLog_SaveEvents">True</setting>
```

Log File:

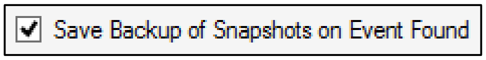


```
"10/5/2015 2:03:38 PM","File SHA512 signature has changed","
"10/5/2015 2:03:38 PM","File SHA512 signature has changed","
"10/5/2015 2:05:38 PM","File SHA512 signature has changed","
"10/5/2015 2:08:12 PM","File SHA512 signature has changed","
```

[redacted]

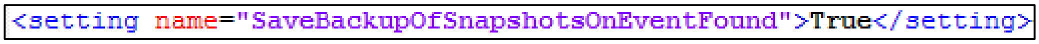
[Local] Backup Snapshot

Configured from the GUI “Update Config” option under “Settings.Config” tab:



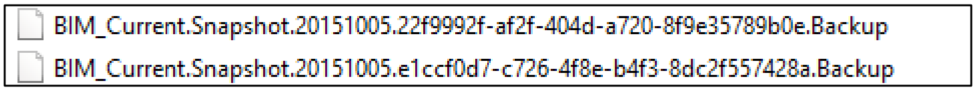
☑ Save Backup of Snapshots on Event Found

Or also updated directly in the Settings.Config file:



```
<setting name="SaveBackupOfSnapshotsOnEventFound">True</setting>
```

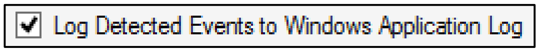
Backup Files:



```
BIM_Current.Snapshot.20151005.22f9992f-af2f-404d-a720-8f9e35789b0e.Backup
BIM_Current.Snapshot.20151005.e1ccf0d7-c726-4f8e-b4f3-8dc2f557428a.Backup
```

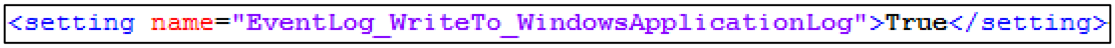
[Windows] Application Log

Configured from the GUI “Update Config” option under “Settings.Config” tab:



☑ Log Detected Events to Windows Application Log

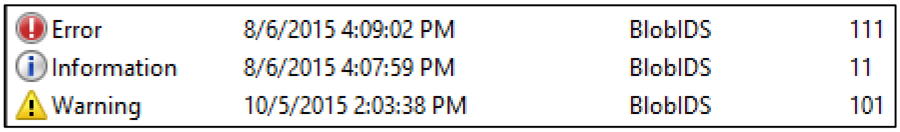
Or also updated directly in the Settings.Config file:



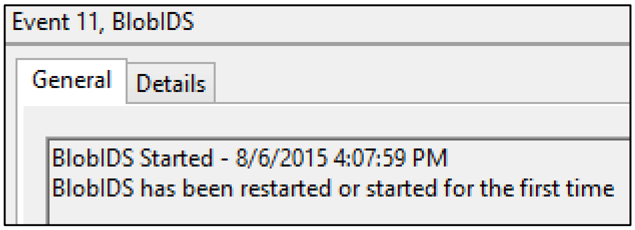
```
<setting name="EventLog_WriteTo_WindowsApplicationLog">True</setting>
```

Application Log:

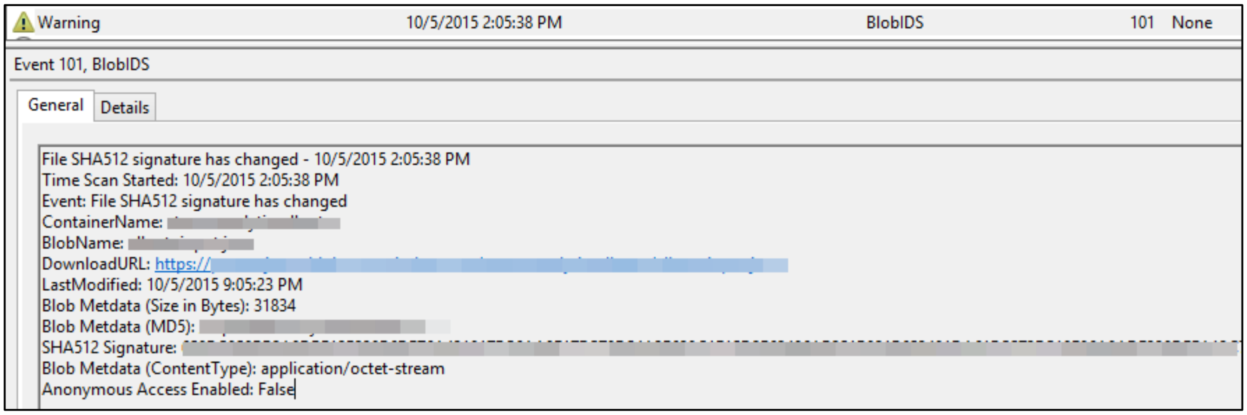
Event ID **111** = Error
Event ID **11** = Informational – Tool Event (e.g BIM restarted)
Event ID **101** = Warning – Event Detected (e.g: blob changed)



Error	8/6/2015 4:09:02 PM	BlobIDS	111
Information	8/6/2015 4:07:59 PM	BlobIDS	11
Warning	10/5/2015 2:03:38 PM	BlobIDS	101



BlobIDS Started - 8/6/2015 4:07:59 PM
BlobIDS has been restarted or started for the first time

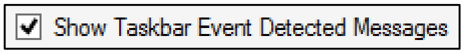


```
Warning    10/5/2015 2:05:38 PM    BlobIDS    101    None
```

```
File SHA512 signature has changed - 10/5/2015 2:05:38 PM
Time Scan Started: 10/5/2015 2:05:38 PM
Event: File SHA512 signature has changed
ContainerName: [redacted]
BlobName: [redacted]
DownloadURL: https://[redacted]
LastModified: 10/5/2015 9:05:23 PM
Blob Metdata (Size in Bytes): 31834
Blob Metdata (MD5): [redacted]
SHA512 Signature: [redacted]
Blob Metdata (ContentType): application/octet-stream
Anonymous Access Enabled: False
```

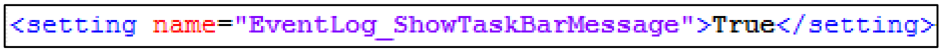
[Windows] TaskBar Notification

Configured from the GUI “Update Config” option under “Settings.Config” tab:



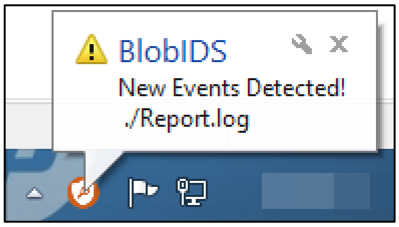
☑ Show Taskbar Event Detected Messages

Or also updated directly in the Settings.Config file:



```
<setting name="EventLog_ShowTaskBarMessage">True</setting>
```

Notification:



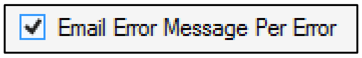
BlobIDS

New Events Detected!
./Report.log

Error Logging

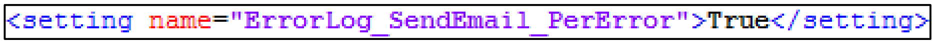
[Email] Alert Per Error

Configured from the GUI “Update Config” option under “Settings.Config” tab:



☑ Email Error Message Per Error

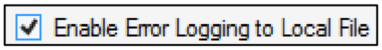
Or also updated directly in the Settings.Config file:



```
<setting name="ErrorLog_SendEmail_PerError">True</setting>
```

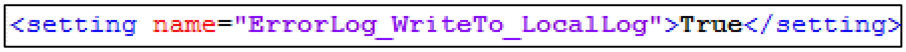
[Local] Log File

Configured from the GUI “Update Config” option under “Settings.Config” tab:



☑ Enable Error Logging to Local File

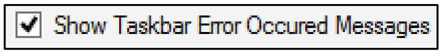
Or also updated directly in the Settings.Config file:



```
<setting name="ErrorLog_WriteTo_LocalLog">True</setting>
```

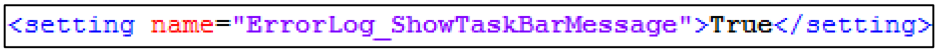
[Windows] TaskBar Notification

Configured from the GUI “Update Config” option under “Settings.Config” tab:



☑ Show Taskbar Error Occured Messages

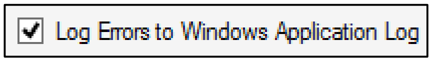
Or also updated directly in the Settings.Config file:



```
<setting name="ErrorLog_ShowTaskBarMessage">True</setting>
```

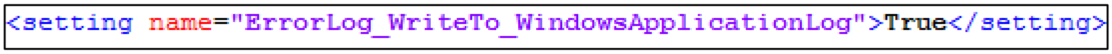
[Windows] Application Log

Configured from the GUI “Update Config” option under “Settings.Config” tab:



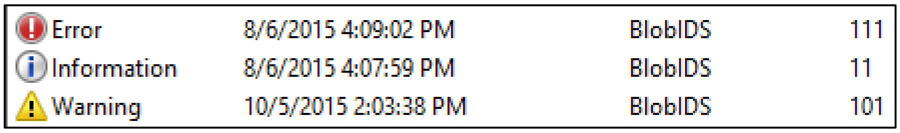
☑ Log Errors to Windows Application Log

Or also updated directly in the Settings.Config file:



```
<setting name="ErrorLog_WriteTo_WindowsApplicationLog">True</setting>
```

Event ID **111** = Error
Event ID **11** = Informational – Tool Event (e.g BIM restarted)
Event ID **101** = Warning – Event Detected (e.g: blob changed)

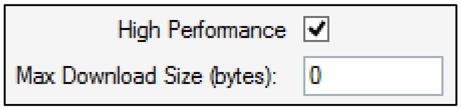


Error	8/6/2015 4:09:02 PM	BlobIDS	111
Information	8/6/2015 4:07:59 PM	BlobIDS	11
Warning	10/5/2015 2:03:38 PM	BlobIDS	101

Performance

High Performance Mode

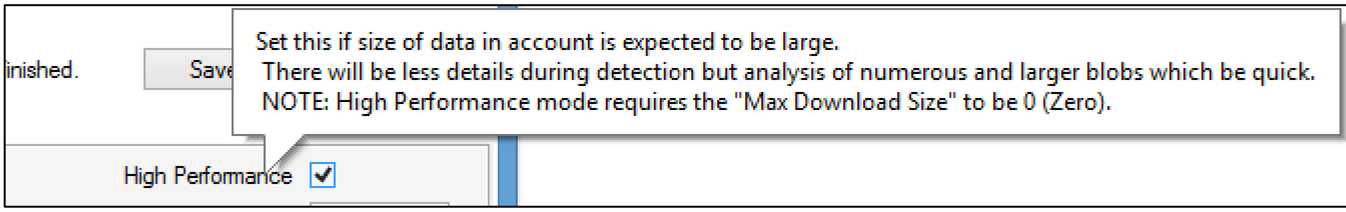
High performance may be needed if frequent scanning intervals are used, such as every minute. Turning on high performance will automatically force the Max Download Size to 0 bytes.



High Performance ☑
Max Download Size (bytes): 0

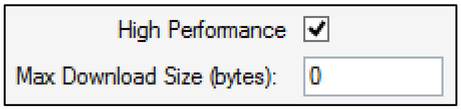
Scanning Performance (Max Download Size)

If each scan into blob storage is taking longer than desired, the “Max Download Size” can be set to 0 bytes. Setting the max download size to zero will ensure that blobs are not downloaded when doing a check/scan, instead the blob’s metadata is queried and used as the sole information for comparison. The primary detection will be around the date modified property of the blob. This could reflect a possible change in the metadata itself or it could be a change in the blob data, specification of which it is from this point is not possible (vs. if a blob is actually downloaded during scanning, BIM will be able to detect if the SHA512 hash changes, this is an indication the actual data in the blob has changed).



Set this if size of data in account is expected to be large.
There will be less details during detection but analysis of numerous and larger blobs which be quick.
NOTE: High Performance mode requires the "Max Download Size" to be 0 (Zero).

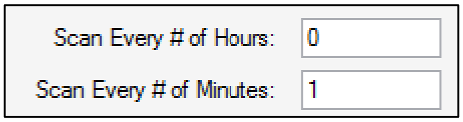
High Performance ☑



High Performance ☑
Max Download Size (bytes): 0

This can also be set manually in the configuration file.

Additionally if the contents of a storage account is large, a quick scanning interval may not be optimal and a larger scanning interval might be a better option.

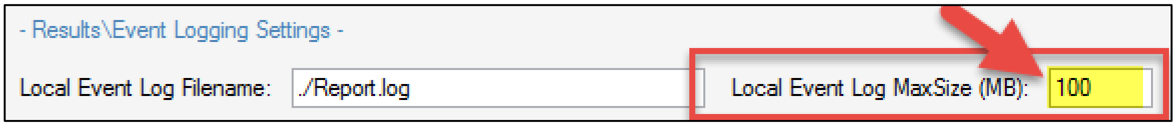


Scan Every # of Hours: 0

Scan Every # of Minutes: 1

Logging Performance

If logs are becoming unwieldly in size, the max size of a log file can be limited.



- Results\Event Logging Settings -

Local Event Log Filename: ./Report.log

Local Event Log MaxSize (MB): 100

Blob Storage Transactions Costs

Note, the content in this posting is subject to change as we add more features to the storage system. This posting is given as a guideline to allow services to be able to estimate their storage bandwidth, transactions and capacity usage before they go to production with their application.

Basic Costs

- Storage transactions (http://azure.microsoft.com/en-us/pricing/details/storage/)
- $0.0036 per 100,000 transactions for all Standard storage types.
- Transactions include both read and write operations to storage.
- There are no storage transaction charges for Premium Storage.

Definitions

When using Windows Azure Blobs, Tables and Queues storage costs can consist of:

- Bandwidth – the amount of data transferred in and out of the location hosting the storage account
- Transactions – the number of requests performed against your storage account
- Storage Capacity – the amount of data being stored persistently

Bandwidth – Since applications need to compute over their stored data, we allow hosted services to be co-located with their storage. This allows us to provide free bandwidth between computation and storage that are co-located, and only charge bandwidth for storage when accessed from outside the location it is stored in.

Transactions – Each individual Blob, Table and Queue REST request to the storage service is considered as a potential transaction for billing. Applications can then control their transaction costs by controlling how often and how many requests they send to the storage service. We analyze each request received and then classify it as billable or not billable based upon our ability to process the request and the request's outcome.

Capacity – We accumulate the size of the objects stored (blobs, entities and messages) as well as their application and system metadata in order to measure the storage capacity for billing.

In the rest of this post we will explain how to understand these three areas for your application.

When Bandwidth is Counted

In order to access Blobs, Tables and Queues, you first need to create a storage account by going to the Windows Azure Developer Portal. When creating the storage account you can specify what location to store your storage account in. The six locations we currently offer are:

US North Central
US South Central
Europe North
Europe West
Asia East
Asia Southeast

All of the data for that storage account will be stored and accessed from the location chosen. Some applications choose a location that is geographically closest to their client base, if possible, to potentially reduce latency to those customers. A key aspect here is that you will want to also choose in the Developer Portal the same location for your hosted service as the storage account that the service needs to access. The reason for this is that the bandwidth for the data transferred within the same location is free. In contrast, when transferring data in or out of the assigned location for the storage account, the bandwidth charges listed at the start of this post will accrue.

Now it is important to note that bandwidth is free within the same location for access, but the transactions are not free. Every single access to the storage system is counted as a single transaction towards billing. In addition, bandwidth is only charged for transactions that are considered to be billable transactions as defined later in this posting.

Another concept to touch on in terms of bandwidth is when you use blobs with the Windows Azure Content Delivery Network (CDN). If the blob is not found in the CDN (or the blob's time-to-live (TTL) has expired) it is read from the storage account (origin) to cache it. When this occurs, the bandwidth consumed to cache the blob (transfer it from the origin to the CDN) is charged against the storage account (as well as a single transaction). This emphasizes that you should use a CDN for blobs that are referenced enough to get cache hits, before they expire in the cache due to the TTL, to offset the additional time and cost of transferring the blob from your storage account to the CDN.

Here are a few examples:

Your storage account and hosted service are both located in "US North Central". All bandwidth for data accessed by your hosted service to that storage account is free, since they are in the same location.

Your storage account is located in "US North Central" and your hosted service is located in "US South Central". All bandwidth for data accessed by your hosted service to the storage account will incur the bandwidth charges listed at the start of this post.

Your storage account is located in "US North Central", and your blobs are cached and served by one of the Windows Azure CDN edge locations in Europe . Since the Windows Azure CDN edge location is not in the same location as your storage account, when the data is read from your storage account to the Windows Azure CDN for caching it will incur the above bandwidth charges.

Your storage account is located in "US North Central" but accessed by websites and services around the world. Since it isn't being accessed from a Windows Azure hosted service within the same location the standard bandwidth charges are applied.

How Transactions are Counted

The first area we would want to cover for transactions is what equals 1 transaction to Windows Azure Storage. Each and every REST call to Windows Azure Blobs, Tables and Queues counts as 1 transaction (whether that transaction is counted towards billing is determined by the billing classification discussed later in this posting). The REST calls are detailed here:

- Blobs
- Tables
- Queues
-

Each one of the above REST calls counts as 1 transaction. This includes the following types of requests:

Query/List Requests and Continuation Tokens – A Table Query, and Listing Blob Containers, Tables and Queues can return continuation tokens. This means that the query/listing must be continued to complete it. As described above, each REST request to the storage service counts as 1 transaction. Therefore, each continuation of the query/list counts as an additional 1 transaction, since it is another REST request to the storage service.

Batch Operations – We currently support two types of batch operations:

- **Get Messages** - the ability to get up to 32 messages at once from a Queue.
- **Entity Group Transactions** – the ability to perform an atomic transaction across up to 100 entities with any combination of Insert, Update, or Delete for Azure Tables. The requirement is that all of the entities have to be in the same table and have the same PartitionKey value and the total request size must be under 4Mbytes.

Both of these types of batch operations result in a single REST request to the storage service. Therefore, they count as a single transaction for each request.
When using the Storage Client Library, there are a few function calls that can result in multiple REST requests to your storage account.

Uploading Blobs – When uploading a blob greater than 32 Mbytes, the storage client library will break it into 4 Mbyte blocks by default. The block size can be changed by setting the CloudBlobClient.WriteBlockSizeInBytes field. When uploading a blob larger than 32MBs the client library will upload each block as a separate PutBlock REST request and then commit all of the blocks at the end with a PutBlockList. Each PutBlock will count as 1 transaction, and the final PutBlockList will also count as 1 transaction.

Table Queries – When you query using CloudTableQuery it takes care of handling continuation tokens, so it reissues queries using the continuation token receieved in a previous query request to get remaining entities. As described above, each reissued continuation token query to the service counts as 1 transaction.

Table SaveChanges – For the Table service, when you do an AddObject, UpdateObject or DeleteObject the entity gets added to a data context so that at some later point the changes can be flushed to your storage account when the program executes SaveChangesWithRetries. When SaveChangesWithRetries executes, all of the pending changes are then pushed to the Table service one at a time each using its own REST call. So each pending change counts as 1 transaction.

The one exception to this is the Entity Group Transaction. If you have batched up operations (AddObject, UpdateObject, DeleteObject) for a set of entities with the same PartitionKey value, you can execute a single SaveChangesWithRetries(SaveChangesOptions.Batch), and then this will send all of the changes to the table service as a single REST request and it will count as 1 transaction. The key is to make sure to pass in the SaveChangesOptions.Batch option to SaveChangesWithRetries method. We have seen services forget the SaveChangesOptions.Batch and this results in each pending change to be sent as a separate request leaving the customer wondering why the SaveChanges is taking a lot longer than it should, the operations would not be atomically performed as a single transaction (meaning they either all commit or none of them commit), and they would end up with a higher transaction count than expected.

Here are a few examples:

A single GetBlob request to the blob service = 1 transaction

PutBlob with 1 request to the blob service = 1 transaction

Large blob upload that results in 100 requests via PutBlock, and then 1 PutBlockList for commit = 101 transactions

Listing through a lot of blobs using 5 requests total (due to 4 continuation markers) = 5 transactions

Table single entity AddObject request = 1 transaction

Table Save Changes (without SaveChangesOptions.Batch) with 100 entities = 100 transactions

Table Save Changes (with SaveChangesOptions.Batch) with 100 entities = 1 transaction

Table Query specifying an exact PartitionKey and RowKey match (getting a single entity) = 1 transaction

Table query doing a single storage request to return 500 entities (with no continuation tokens encountered) = 1 transaction

Table query resulting in 5 requests to table storage (due to 4 continuation tokens) = 5 transactions

Queue put message = 1 transaction

Queue get single message = 1 transaction

Queue get message on empty queue = 1 transaction

Queue batch get of 32 messages = 1 transaction

Queue delete message = 1 transaction

Now that we understand what a transaction is, let’s describe what transactions are counted towards billing and what transactions are not counted.

Transactions Not Counted

When a transaction reaches our service, if it falls into one of the following classifications we do not count it towards billable transactions, and no bandwidth is charged for these transactions:

Pre-Authentication Failure – If we do not even get the chance to authenticate the transaction then it does not count towards billable transactions. Examples here include malformed requests with bad http headers, badly formed URLs, the time range for the request is invalid, etc.

Authentication Failure – if the transaction fails authentication we do not count it towards the billable transactions for the storage account.

Quota Permission Failure – we have a 100TB quota per storage account. If a storage account hits that limit, then we put the storage account into a mode where it has only READ+DELETE permissions, but no WRITE permissions. This allows the storage account to perform Get and Delete operations, but not put/post operations. When in this mode, requests that require write permissions will fail, and we do not count these towards billable transactions.

Incorrect Shared Access Signature (SAS) HTTP Verb/Permission – If the sent signature correctly validates (authenticates), but the REST operation keyword (PUT, POST, GET, DELETE) does not correctly match the permissions, then the request will not be counted towards billable transactions. For example, if the permissions specify that only PUT can be performed, but a GET verb is used with a valid SAS, the request will fail and not be counted towards billable transactions.

Anonymous Request Failures – If a request comes in without a signature it will be treated as an anonymous request, and we classify the following 3 types of failures as transactions that do not count towards billable transactions:
Incorrect Permissions – Anonymous requests can only be GET requests. If it is not a GET request, it is rejected and does not count towards billable transactions.
Container Not Found – If the container does not exist for the anonymous GET request, then it is not counted towards billable transactions.
Blob Not Found – If the blob trying to be accessed does not exist for the anonymous GET request, then it is not counted towards billable transactions.

Unexpected Timeouts – If the request times out due to an issue in the storage system, it is not counted towards billable transactions.

If any of the above conditions apply then the transaction is not counted towards billable transactions and no bandwidth is charged for the request. Then for the rest of the transactions we classify them as billable and they may or may not incur bandwidth charges as described in the bandwidth section.

Transactions Categorization

We categorize the billable transactions into the following buckets:

Successful Transactions – Any successful transaction completed against the storage system.
Expected Failures– The expected failures come from the following three areas:
Transaction Errors – These are requests that are correctly authenticated with correct permissions that fail due to the transaction semantics being applied against the data in the storage account. Some examples are:

- Trying to get or delete an object that has already been deleted or never existed, which results in a NotFound error.
- Trying to create an object that already exists, which results in an AlreadyExists error. For example, we have seen applications that perform a CreateIfNotExist on a Queue before every put message into that queue. This results in two separate requests to the storage system for every message they want to enqueue, with the create queue failing. Make sure you only create your Blob Containers, Tables and Queues at the start of their lifetime to avoid these extra transaction costs.
- Doing a conditional operation with an ETag Match, Non-Match, Modified-Since, or Unmodified-Since, and having the operation fail (getting back a NotModified or PreconditionFailed) will count as a transaction.
- Trying to add a Table Entity that already exists will result in a failure (Conflict - 409). Similarly, trying to Update an entity that does not exist will result in a failure (Not Found - 404). These count towards transaction costs. This is actually an issue for applications that want to do an Upsert, which we

are currently evaluating how to support for Windows Azure Tables. Until Upsert is provided, users should evaluate their usage scenario to decide if they should either do INSERT before UPDATE or UPDATE before INSERT in order to minimize the amount of expected failures and overall transactions to Windows Azure Tables.

- Valid Shared Access Signature (SAS) with a ContainerNotFound or BlobNotFound. If the SAS signature validates correctly, but the container is not found or the blob being accessed is not found, then this is counted as a billable transaction.
- Etc. The list is quite long, since there are many ways requests can encounter an expected failure based on the semantics exposed by the storage service (e.g., conditional operations, leases, sequence numbers, etc).

Throttling – These are requests that are being throttled due to the transaction rate going over the per partition target throughput described in the post "Windows Azure Storage Abstractions and their Scalability Targets". These throttled requests are counted as billable transactions. When this occurs, the client is expected to use exponential backoff and retry the request, which is provided by default with the storage client library. If it is a reoccurring event for the service, then the service should consider additional partitioning of its data structures as described in the upcoming posts on Blobs, Tables and Queues.

Expected Timeouts – When you send a request to the service you can specify your own timeout and you can set it to be smaller than the SLA timeout. If the request has a timeout less than the SLA timeout, and the request times out, it is classified as an expected timeout and counted towards billable transactions.