ENF Monitoring Suite – Get-ENFAlertsKyle Timins
Created 2016Kyle Timins

ENF Alert Emails

May 16, 2017

Contents

List of Figures

List of Tables

Get-ENFAlerts

The Get-ENFAlerts is a tool used to aggregate and alert when ENFs occur in the specified environments. It has been developed in Powershell v3.0 and JSON. It works in conjunction with configuration files, written in JSON, that are passed in via command line arguments.

Config Files

The configuration files that are used via this script are written in JSON. JSON is an Object Notation Language created from Javascript. JSON was chosen because it is light weight, easy to parse, and easier to read when compared to XML. If not familiar with JSON, a quick tutorial can be found at the following link: http://www.tutorialspoint.com/json/json _ quick _ guide.htm

There are two configuration files that are passed into the script: the enfrc and the envrc.

enfrc.json

The enfrc.json file is the configuration file that is used to define the parameters of the current " run " of the script. This includes what groupings of ENF messages should be looked at, what environments should be looked at, and the configuration of the email that will be sent out. It can be configured to look at multiple environments and/or ENF exception sets so an aggregated list of everything will be sent out. Or it can be configured so there are multiple versions of the configuration file so each only looks at one environment and one set of ENF exception messages, thus creating multiple emails (via multiple runs) that are segmented as to be specific in nature. As a user will tend to end up with a lot of enfrc configuration files, it is recommended to give them proper names, but keep the " enfrc.json " bit at the end to differentiate from the envrc configuration file.

By default, the script looks at the first unnamed parameter or the named parameter " configFile " . If no parameter is passed in for the enfrc configuration file, it will default to looking for a file called " enfrc.json " in the same directory level as the Get-ENFAlerts.ps1 file.

Structure and Contents

- **Output**: This contains the information for the output of the script including files created and the email. It is an object.

- **File**: This contains the information for the files that can be created from this script. It is an object.
  - **CSV**: Determines if CSV files should be created with data that gets put into the email. It is a Boolean.
  - **JSON**: Determines if JSON files should be created with data that gets put into the email. It is a Boolean.
  - **XML**: Determines if XML files should be created with data that gets put into the email. It is a Boolean.
  - **Zips**: A global on/off switch to determine if the zip files should be downloaded for the Triage ENFs. It is a Boolean.
  - **Suffix**: This is the suffix that will be added to the end of the CSV/JSON/XML files created using this config file. It is a string.
  - **AppendDate**: Determines if a datetime stamp should be added to the folder, in which, the files are saved to, if they are to be saved. It is a Boolean.
  - **Save**: Determines if the CSV/JSON/XML files should be saved to the local disk or if they should be deleted after they have been sent in the email. It is a Boolean.
- **Email**: This is the configuration for the email settings. It is an object.
  - **Subject**: This is the subject of the email that will be sent out. It is static and has no escape characters. It is a string.
  - **Recipient**: This is a hard coded list of recipients for the email. It is an array of objects.
  - **RecipientsScheduled**: This is the way to dynamically add recipients to an email based on a start/end date pair or a service pack. It is an array of objects.
    - **Name**: The name of the person. It is a string.
    - **Email**: The email of the person. It is a string.
    - **ServicePack**: List of service packs that the person should be added to the email. Service packs are counted from Monday to the Friday two weeks ahead. This has priority over the Dates. It is an array of strings.
    - **Dates**: If not using the service pack, the date list holds the start and end dates. It is an array of objects.

        - ▹ **StartDate**: The start date of when the user should be added as a recipient. It is a string.
        - ▹ **EndDate**: The end date of when the user should be added as a recipient. This has priority over the TimeSpan field. It is a string.
        - ▹ **TimeSpan**: The Timespan that the user should be added as a recipient. Use either this or the end date field. The format for this is m/ \ d+[mhd]/ meaning a number followed by 'm' for minutes, 'h' for hours, and 'd' for days. It is a string.
    - ∗ **CC**: This is a hard coded list of carbon copy recipients for the email. It is an array of strings.
    - ∗ **Attachment**: Should the CSV, XML, and JSON files be attached to the email. It is a Boolean.
    - ∗ **Priority**: What should the priority of the email be? High, normal, or low are the valid entries.

- **BaseDir**: This is where all the local files are created and stored, if the settings say so.

- **ENFSets**: This is the list of ENF sets that the script should check against. It is an array of objects.

  - ◦ **Name**: This is the name of the set that will be used by the script. It is a string.
  - ◦ **TimeSpan**: This is how long the script should look back for the ENFs in this set. The format for this is m/ \ d+[mhd]/ meaning a number followed by 'm' for minutes, 'h' for hours, and 'd' for days. It is a string.
  - ◦ **Zips**: This is how many zips should be downloaded for the set if the setting above says to download the zips. This may become depricated. The format for this is m/ \ d+[fl]/ meaning a number followed by either 'f' for first or 'l' for latest zips. It is a string.
  - ◦ **MinCount**: This is the number of occurrence enfs per Triage ENF that should occur before the script will count it the numbers or the email. It is an integer.
  - ◦ **Email**: These are the email settings for the ENF set. They determine how the ENF displays on the email. It is an object.
    - ∗ **Threshold**: Threshold of how many occurrences are need to occur in the set for it to be added to the email. '0' means always send an email. It is an integer.

    * **Details**: Should the details section be shown for the ENFs in this set. Contains useful information, but adds bulk to the email. This controls the whether the rest of the settings in Email matter or not. It is a Boolean.
    * **Stack**: Should the stack be shown in the email section. Can make the email very bulky. It is a Boolean.
    * **OrderBy**: This is the setting to show what order the details section of the email should be ordered in. The valid values are " count " and " date " . Both are descending.
    * **IncludeNotes**: Should the notes on the ENF be shown in the email. These are the notes that are added to the ENF in the ENF tool. It is a Boolean.
    * **IncludeEmptyNotes**: Should there be an empty section for the recipients to add their notes to. This may be depricated. It is a Boolean.
    * **IncludeLatestInfo**: Should there be information in the details section for the latest occurrence. It is a Boolean.
  - **ExMsgs**: List of exception messages that should be looked at by the script for this set. It is an array of strings.
  - **NotExMsgs**: List of exception messages that should not be looked at by the script for this set. It is an array of strings.

- **ENFPortfolio**: This is the portfolio that the script should narrow the search to. Used for Daily Builds LOB teams. It is an array of strings.

- **ENFSkippedPortfolio**: This is the set of portfolios that the script should skip when running it's search. It is an array of strings.

- **ENFSkippedStatuses**: This is the list of status of an ENF that should be skipped when getting the list of ENFs. It is an array of strings.

- **Envs**: This is the environments that should be looked at for each ENFSet. It is an array of objects.

  - **Client**: This is the client of the environment. The spelling and correct values is determined by the environmental configuration file. It is a string.
  - **Env**: This is the environment within the client. The spelling and correct values are determined by the environmental configuration file. It is a string.

---

envrc.json

The envrc.json file is the environmental configuration file that is used to define the environment that the script is running in. It includes info like the ENF Database server, the SMTP server, the list of valid clients with their environments (and the ENF database tables that store their ENFs), and the conversion from Service Packs to weeks. This configuration will most likely not change between different " runs " , so it is mostly safe to keep one with just the name " envrc.json " . If more are needed, keep the " envrc.json " bit at the end to differentiate from the enfrc configuration file.

By default, the script looks at the third unnamed parameter or the named parameter " envConfigFile " . If no parameter is passed in for the envrc configuration file, it will default to looking for a file called " envrc.json " in the same directory level as the GetENFAlerts.ps1 file.

Structure and Contents

- **ServicePacks**: This is a list of service packs with their start and end dates. It is used in the script to determine if a scheduled recipient should be added to the recipients or not for a desired alert. It is an array of objects.
    - **Release**: Service pack release name. It is a string.
    - **StartDate**: The start date of the service pack. This is the Sunday of the first week. It is a string.
    - **EndDate**: The end date for the service pack. This is the Saturday of the second week. It is a string.
- **ENVLookup**: This is the list of clients with their valid environments and what SQL database to look at. It is an array of objects.
    - **Client**: This is the name of the client. It is a string.
    - **ENVs**: This is a list of environments the client has and what SQL database to look at. It is an array of objects.
        - **Name**: This is the environment that the client has. It is a string.
        - **SQLDB**: This is the SQL database in the ENF SQL Server to look at for the ENFs. It is a string.

- **DBServer**: This is the SQL Database server to use in the DB lookups. It is a string.
- **SMTPServer**: This is the SMTP server to use when sending the email. It is a string.

How to Set Up and Customize the Configuration Files

This section will show how to set up the enfrc configuration file for a customized run. *The Bare Essentials* will show the most minimalized setup needed to be ready to run the script. The *Full Script Configuration* will show most of the different configurations that are possible to completely customize the enfrc configuration file. See above section, Structure and Contents , for more information on the enfrc configuration file is structured and what it contains. Note: This section is only about customizing the enfrc configuration file. It does not cover running the script or setting up Scheduled Tasks. These are explained in Running Get-ENFAlerts.ps1 and How to Run Get-ENFAlerts.ps1 via Scheduled Tasks .

The Bare Essentials This section will show what the bare, minimal essentials are needed to set up the enfrc configuration file for customized run.

- What LOBs should the script match against?
  - These are put in the " ENFLOBs " section, which is a comma separated array. The LOB codes should be the two letter codes that are used in the ENF tool for the Line of Business. (Should be the same codes that are used in Policy Decisions.) E.g. " GL " , " PF " ,. . .
- Who should be the recipients and the CC recipients?
  - Recipients are put in a comma separated array under *Output.Email.Recipient*. By default, the email address associated with the account used to run the script will be automatically added to the Recipients array when it sends out the email.
  - Carbon Copy Recipients are put in a comma separated array under *Output.Email.CC*.
- What Exception message(s) should the script compare against?
  - By default, the script comes with an ENF set that looks against the Exception Message " % " . This means " match any and every exception message " . Unless specificity is needed, this should be sufficient for running this script. These are stated in a comma separated array under *ENFSets. _ object _ .ExMsgs*.

- What ENF statuses should be skipped?
    - By default, the script will skip " Deferred " , " Environmental " , " Resolved " , and " In Progress " . If desired, In Progress may be removed from the list to show any ENFs that have this status. It is not recommended to remove any of the other three statuses as they are meant to remove the ENFs from view. These are stated in a comma separated array under *ENFSkippedStatuses*.
- What Client and Environments should the script look at?
    - By default, the script is setup to look at DailyBuilds (Prod), Allstate (Prod), and Allstate (QA). If running for Daily Builds only, it is recommended that the two references for Allstate are removed. The valid Client/Environment pairs are specified in the envrc.json configuration file. These are stated in *Envs. _ object _* .
- What should the threshold of ENF occurrences be?
    - The threshold is the number of ENF occurrences in an ENF set that need to happen in the given timespan before an email is sent out. A threshold of 0 means always sends an email, even if there are no occurrences. A threshold of 1 or more means send an email, only if the total number of ENF occurrences in the set is equal to or greater than the threshold. By default, the threshold is set to 1. This can be found under *ENFSets. _ object _ .Email.Threshold*.
- What time frame should be looked back at?
    - By default, the script is setup to look back two hours for the ENF set. This is great for monitoring emergency issues, but for standard use, it is recommended to look back between one day and seven days. (Or one hundred days for the exteme.) It is recommended that the run at the end of the week should look back over the entire week, if not further. And possibly overlapping any daily or every X number of days. (Like running thirty hours back instead of one day/twenty-four hours.) This is because there can be a delay between the error occurring and the ENF being created.

Full Script Configuration Please see the default enfrc.json and envrc.json files to see the full extent of the customizability of the configuration files for this script.

The Script

The Get-ENFAlerts.ps1 script is the main tool for this as it contains the code of the program. It is built upon Powershell v3 and.Net technology. As such, .Net 3.5 and Powershell v3 must be installed on the machine prior to running the script.

Parameters

The script takes in four optional parameters:

- enfrc.json configuration file
- Starting directory
- envrc.json configuration file
- Validate only switch.

The configuration files are explained above.

The Starting directory defines the starting directory where the paths for the configuration files will be found. The starting directory can be used if running from a tool like " Task Scheduler " as it will start the powershell instance in " C: \ Windows \ System32 " . Or it can be used if the script is run from a different directory as the config files.

The Validate Only switch can be used to make sure that the configuration files are in the right format. It will not run any database lookups or send the email; it will only validate the two configuration files. It is recommended that this option be run after any changes to the configuration files.

Functionality

The script follows a set procedure in how it works and can be understood by that. This is the process in which the script works:

1) The enfrc and envrc files are passed into the script when it is run.
2) The script processes the config files and turns them into smaller object that it uses to hold the configuration information for each part.
3) The Base Directory and other high level variables are set.

4) Foreach ENF set that is in the configuration file:

    a) It creates an object for that set.
    b) It goes out and gets a list of Triage ENFs that have occurred within the specified parameters of the set.
    c) Foreach of the Triage ENFs the last section returned:

        (i) It goes out and gets the latest occurrence information for the ENF
        (ii) It goes and gets the occurrence count for the Triage ENF within the time frame supplied.
        (iii) An object is created with the information about the Triage ENF, including the occurrence information. This object is then added to the ENF Set object.
        (iv) It is determined if the mail threshold has been fulfilled to determine if an email should be sent.

    d) If the mail threshold has not be fulfilled, it checks to see if it should be sending an email based on the threshold being set to 0.
    e) Add Set object to an " Outset " array.

5) If specified by the configuration file states, the CSV, XML, and/or the JSON files are created and saved to the working directory within the Base Directory.

6) The Email is created by going through the sets.

    a) The email is only created and sent if the threshold is reached or the threshold for the set is set to 0.
    b) The summary section is created to give a high level overview.
    c) The details section is then created to show more in-depth detail about each ENF.
    d) The script goes through the working directory to add any attachments to the email that have been created.
    e) The email is then sent.

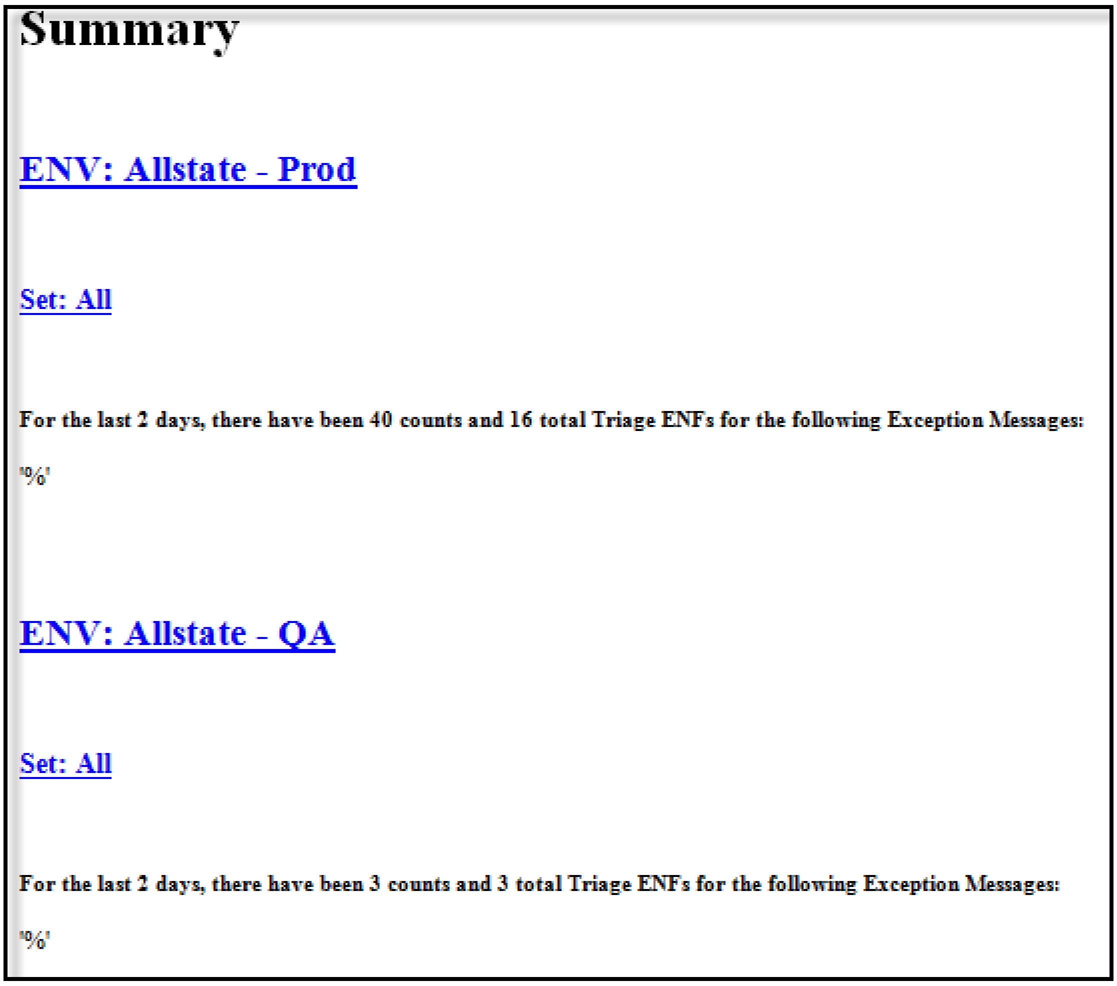
7) If the files are set to be saved, the files are then moved to their permanent place in the Base Directory. Else, the files are deleted from the working directory.

Output

There are three types of outputs that can be created by this script: the email, output files, and zip files.

The Email The ENF Alert Message is structured into three main parts: The Summary, the Details, and the Attachments.

Summary The Summary section is a main overview of the ENFs. It includes the environment (ENV) and the ENF set (Set) for that environment. Within each pair, there is a line that states the time length, total occurrences, and number of Triage ENFs for each pair. It also includes the exception message for that pair. (‘ % ’ means that all exception messages are included.) The ENV and Set lines are hyperlinks that will take you to that pair’s Details section. (This works in Outlook on the computer, but will not work on a mobile phone.) Here is an example of a summary with two ENVs and one Set per ENV.



Summary

ENV: Allstate - Prod

Set: All

For the last 2 days, there have been 40 counts and 16 total Triage ENFs for the following Exception Messages:

'%'

ENV: Allstate - QA

Set: All

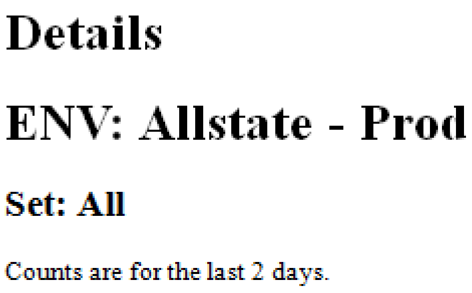
For the last 2 days, there have been 3 counts and 3 total Triage ENFs for the following Exception Messages:

'%'

Figure 1: Figure - Example Summary Section

Details The Details section is also split up into ENV and Set pair. The pairs show the ENV and the Set, along with a line saying how long of a time period was looked at. (As stated above, clicking the Summary ENV or Set name will bring you to its equivalent in the details section.) This section has a table for each Triage ENF for that Pair. This section has the majority of the needed information on the ENF including the GUID, Counts, ENF Status, CSR (if applicable), Exception Message, Stack Trace (if applicable), ENF notes (if applicable), information about the last occurrence, and a link to the latest zip file. (Note: the link to the latest zip file will only work if you are on Insurity's network.) The ENF Notes section will only show if the ENF contains notes in the database/tool.

Here is what a pair header looks like:



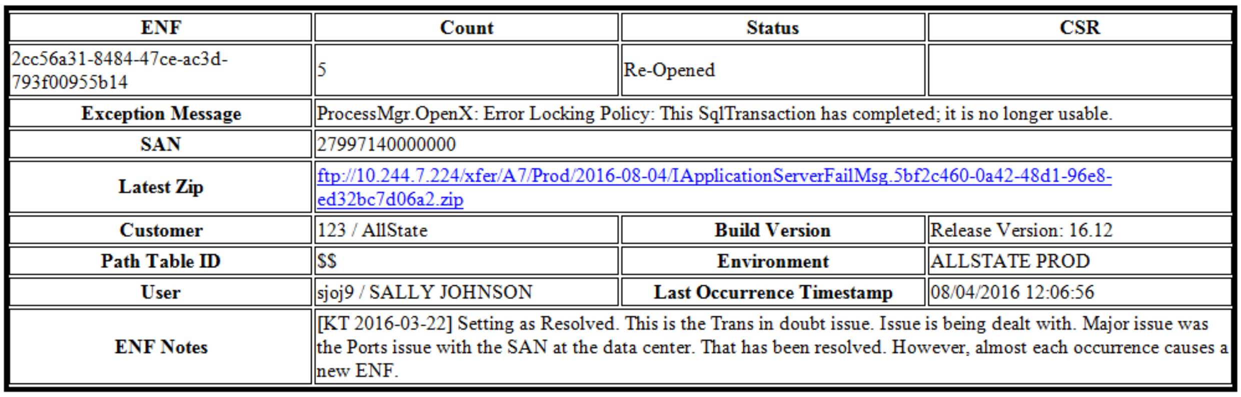
Details

ENV: Allstate - Prod

Set: All

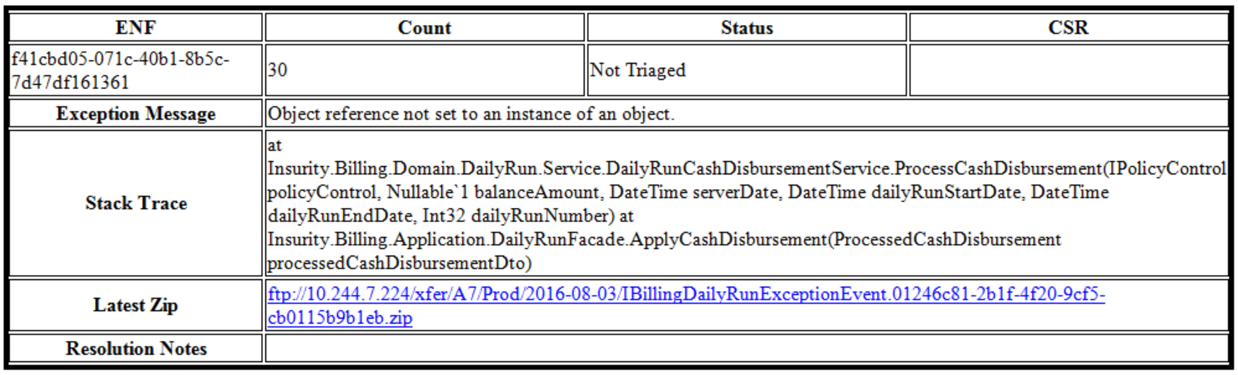
Counts are for the last 2 days.

Here are some examples of what the Details grid might look like:



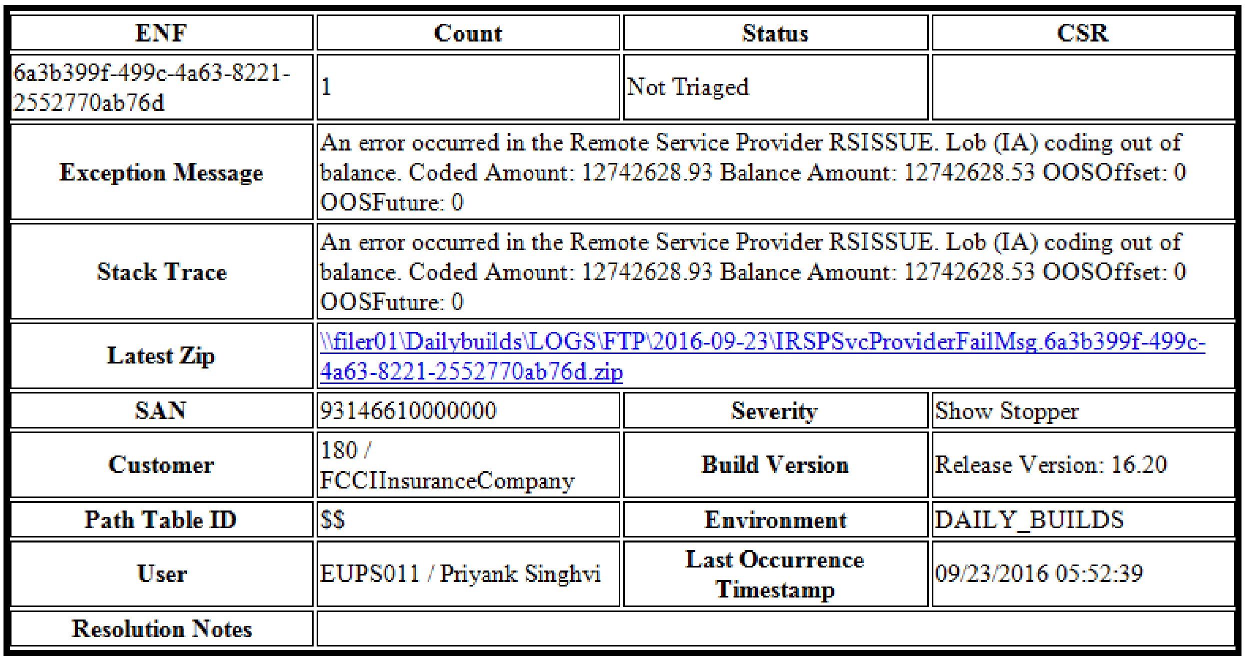
ENF	Count	Status	CSR
2cc56a31-8484-47ce-ac3d-793f00955b14	5	Re-Opened	
Exception Message	ProcessMgr.OpenX: Error Locking Policy: This SqlTransaction has completed; it is no longer usable.		
SAN	27997140000000		
Latest Zip	ftp://10.244.7.224/xfer/A7/Prod/2016-08-04/IApplicationServerFailMsg.5bf2c460-0a42-48d1-96e8-ed32bc7d06a2.zip		
Customer	123 / AllState	**Build Version**	Release Version: 16.12
Path Table ID	$$	**Environment**	ALLSTATE PROD
User	sjoj9 / SALLY JOHNSON	**Last Occurrence Timestamp**	08/04/2016 12:06:56
ENF Notes	[KT 2016-03-22] Setting as Resolved. This is the Trans in doubt issue. Issue is being dealt with. Major issue was the Ports issue with the SAN at the data center. That has been resolved. However, almost each occurrence causes a new ENF.		

Figure 2: Figure - Example Details grid with occurrence information



ENF	Count	Status	CSR
f41cbd05-071c-40b1-8b5c-7d47df161361	30	Not Triaged	
Exception Message	Object reference not set to an instance of an object.		
Stack Trace	at Insurity.Billing.Domain.DailyRun.Service.DailyRunCashDisbursementService.ProcessCashDisbursement(IPolicyControl policyControl, Nullable`1 balanceAmount, DateTime serverDate, DateTime dailyRunStartDate, DateTime dailyRunEndDate, Int32 dailyRunNumber) at Insurity.Billing.Application.DailyRunFacade.ApplyCashDisbursement(ProcessedCashDisbursement processedCashDisbursementDto)		
Latest Zip	ftp://10.244.7.224/xfer/A7/Prod/2016-08-03/IBillingDailyRunExceptionEvent.01246c81-2b1f-4f20-9cf5-cb0115b9b1eb.zip		
Resolution Notes			

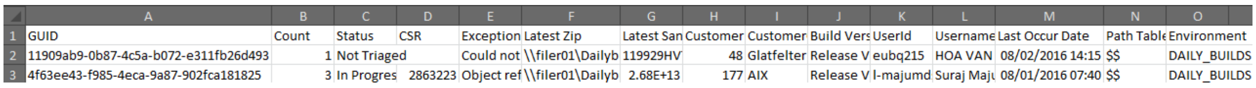
Figure 3: Figure - Example Details Grid with minimal information



ENF	Count	Status	CSR
6a3b399f-499c-4a63-8221-2552770ab76d	1	Not Triaged	
Exception Message	An error occurred in the Remote Service Provider RSISSUE. Lob (IA) coding out of balance. Coded Amount: 12742628.93 Balance Amount: 12742628.53 OOSOffset: 0 OOSFuture: 0		
Stack Trace	An error occurred in the Remote Service Provider RSISSUE. Lob (IA) coding out of balance. Coded Amount: 12742628.93 Balance Amount: 12742628.53 OOSOffset: 0 OOSFuture: 0		
Latest Zip	\\filer01\Dailybuilds\LOGS\FTP\2016-09-23\IRSPSvcProviderFailMsg.6a3b399f-499c-4a63-8221-2552770ab76d.zip		
SAN	93146610000000	**Severity**	Show Stopper
Customer	180 / FCCIInsuranceCompany	**Build Version**	Release Version: 16.20
Path Table ID	$$	**Environment**	DAILY_BUILDS
User	EUPS011 / Priyank Singhvi	**Last Occurrence Timestamp**	09/23/2016 05:52:39
Resolution Notes			

Figure 4: Figure - Example Detail Grid with a CSR

Attachments The last section is the attachments. This contains CSV, XML, and JSON files with the information included in the Details section. But can be parsed via an automated tool (if desired) or used in reports. The default title is formatted as: " ENV-Set—Suffix " , where " Set " is the Client and name of the ENF set and " Suffix " is the name of the group. (Such as " Property " for the Property team.) For example, a title might be " DailyBuilds-Prod-All—Property.csv " . These can be changed if desired. Here is an example of the information, and what it looks like, in the CSV file.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	GUID	Count	Status	CSR	Exception	Latest Zip	Latest San	Customer	Customer	Build Vers	UserId	Username	Last Occur Date	Path Table	Environment
2	11909ab9-0b87-4c5a-b072-e311fb26d493	1	Not Triaged		Could not	\\filer01\Dailyb	119929HV	48	Glatfelter	Release V	eubq215	HOA VAN	08/02/2016 14:15	$$	DAILY_BUILDS
3	4f63ee43-f985-4eca-9a87-902fca181825	3	In Progres	2863223	Object ref	\\filer01\Dailyb	2.68E+13	177	AIX	Release V	l-majumd	Suraj Maju	08/01/2016 07:40	$$	DAILY_BUILDS

Figure 5: Figure - An Example CSV file

The Output Files As mentioned in the Email section, there are some files that are created by the script that can be emailed or saved to disk. These files contain all the relevant information on the ENFs, but in different forms. The output file types are CSV, XML, and JSON. They all contain the same information, but gives the user the ability to get the info in different formats for what they may need them for. This allows the user to run automated reports on the ENF data without having to transform the data into another format.

The creation of these files are determined by Boolean flags in the File.Output section of the enfrc configuration file. As well, that section contains a Boolean value to determine if the files are saved to the computer or if they are deleted after the email is sent.

If the files are to be saved on the computer, they will be saved under a folder created in the Base Directory that is defined in the enfrc configuration file.

The Zip Files The zip files are the ENF zips that are created when an ENF is registered in the system. They include information including the logs, XML messages, polprints, and other useful information. By default, a link to the latest zip is included in the output files and the details section of the email. However, this option allows for the downloading of multiple zips per ENF to the computer. However, due to the size of these files, they are not sent in the email. So it is recommended to only download the zip files if the script is run on a local machine.

There is a " master " Boolean switch in the Output.File section of the enfrc configuration file that controls whether any zips will be downloaded. Then in each of the ENFsets, there is a field to tell the system how many zips should be downloaded per ENF in the set and whether they should from the latest occurrences or the first occurrences.

How to Set Up Powershell

If you have not used Powershell before, this is a quick tutorial on how to setup Powershell to be used to run the Get-ENFAlerts script.

Upgrading to Powershell v3

By default, Windows 7 comes with Powershell v2 installed on the machine. It needs to be upgraded to version three to work with the Get-ENFAlerts script.

1) Ensure that you have Microsoft .NET Framework 4 or higher installed on your machine. If you don't, you can download .NET Framework 4 from the following Microsoft link. Or you can download the latest version of .NET Framework from Microsoft.

    a) https://www.microsoft.com/en-us/download/details.aspx?id=17851

2) Download and install the Windows Management Framework 3.0 (i.e. Powershell v3). It can be found at the following link. Make sure you pick the correct architecture for your computer (x86 vs x64).

    a) https://www.microsoft.com/en-us/download/details.aspx?id=345950x0

3) After you have installed both of these, restart your machine.

4) After you have restarted, open a Powershell window and type the following into the window " $ PSVersionTable " and hit enter. The output should show " PSVersion " being set to 3.0.

How to Setup the Profile file

Execution Policy

To run scripts in Powershell, the execution policy must be changed. The execution policy determines what scripts can be run on Powershell based on where they originated from and whether they are signed. There are five main execution policies:

- Restricted: Does not load configuration files or run scripts. This is the default setting.
- AllSigned: Requires that any and all scripts and configuration files be signed by a trusted publisher, even if the script was written on the computer running it.
- RemoteSigned: Requires that all scripts and configuration files that have been downloaded from the internet be signed by a trusted publisher. All scripts written on that specific machine can be run on that machine without signing the script.
- Unrestricted: Loads all configuration files and runs all scripts, no matter their origin. If running an unsigned script downloaded from the internet, you will be prompted for permission before running it.
- Bypass: Nothing is blocked and there are no warnings or prompts.

Setting Execution Policy in the Profile

For the sake of running the Get-ENFAlerts script, we are using either RemoteSigned or Unrestricted. (Substitute Unrestricted for RemoteSigned in the following if you prefer to use that. This just means you will need to add the certificate to your computer and ensure the script is signed before you run it.)

1) Open Powershell as Administrator.
2) Type the following, sans quotes, into the Powershell window and hit enter. If it prompts you, type " y " and hit enter.

   ```
   Set-ExecutionPolicy Unrestricted
   ```
3) Type the following, sans quotes, into the Powershell window and hit enter.
   a) " New-item –type file –force $ profile "
4) Type the following, sans quotes, into the Powershell window and hit enter. An instance of notepad should open.
   a) " Notepad $ profile "

5) Put the following lines into the file as the first three lines and save the file. (Type them. Do not copy and paste.) Close out notepad.

```
Set-ExecutionPolicy Unrestricted

$ ProfileRoot = (Split-Path –Parent $ MyInvocation.MyCommand.Path)

$ env:path += " ; $ ProfileRoot "
```

6) In the Powershell window, type " . $ profile " . There should be no errors.

How to Import Certificates

Depending on the Execution Policy of the machine, the script may need to be signed. This means that the creator uses a certificate to sign the script with a cryptographic signature at the bottom of the script. It is to show that the script has not been modified since it was signed to avoid tampering and malicious code being added. If the script is signed, the public certificate must be added to the computer's certificate store so that it can validate that the signed script is, indeed, correct. This is only needed on an Execution Policy of AllSigned or RemoteSigned. Unretricted or Bypass do not need a script to be signed.

The certificate for any scripts created by Kyle Timins can be found at " \ \ Hfdk-timinsw7d \ ENF _ Scripts \ PS _ TiminsKy _ Trusted _ Root _ Cert.cer "

How to Add a Public Certificate to a Computer's certificate store

1) Open MMC. This can be done by running " mmc.exe " .

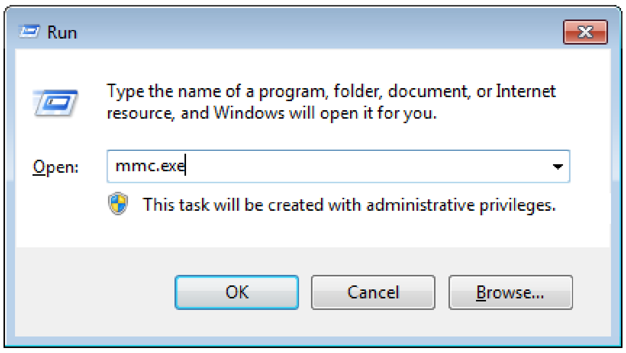
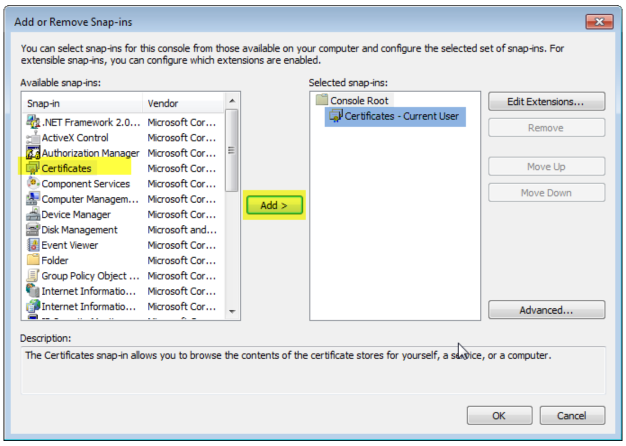

Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
mmc.exe
This task will be created with administrative privileges.
OK
Cancel
Browse...

Figure 6: Figure 2-1 - Run MMC.EXE

1) Click File->Add or Remove Snap-ins

Select ” Current user ” on the screen and hit ” OK ” .

Select ” Certificates ” in the left pane, hit the ” Add > ” button in the middle. You should now see ” Certificates – Current User ” in the right pane. Hit ” OK ” .



Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
.NET Framework 2.0...	Microsoft Cor...
ActiveX Control	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...
Component Services	Microsoft Cor...
Computer Managem...	Microsoft Cor...
Device Manager	Microsoft Cor...
Disk Management	Microsoft and...
Event Viewer	Microsoft Cor...
Folder	Microsoft Cor...
Group Policy Object ...	Microsoft Cor...
Internet Informatio...	Microsoft Cor...
Internet Informatio...	Microsoft Cor...

Selected snap-ins:

Console Root
- Certificates - Current User

Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

Figure 7: Figure 2-2 - Add Certificates Snap-in

1) In the left pane, expand ” Certificates – Current User ” . Then expand ” Trusted Root Certification Authorities ” . Right click on ” Certificates ” , and click ” All Tasks ” then ” Import ” .

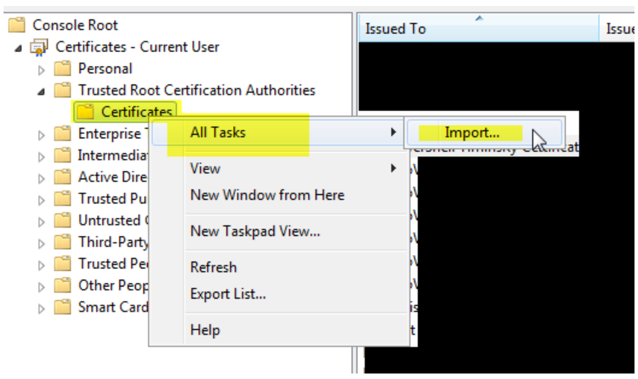
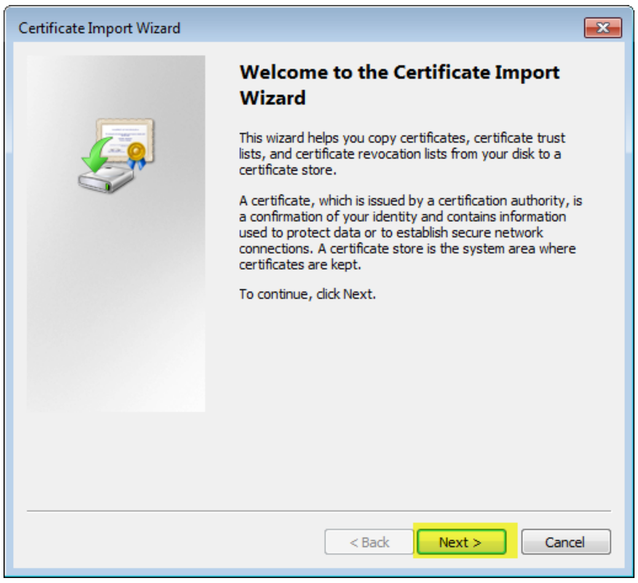


Figure 8: Figure 2-3 - Select Import

1) In the popup " Certificate Import Wizard " , click " Next > "


Certificate Import Wizard

Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

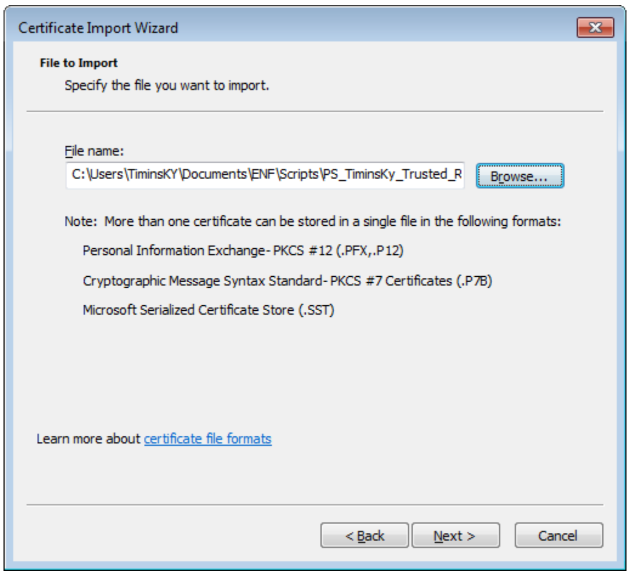
A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

Figure 9: Figure 2-4 - Start Import Process

1) In the File Name text box, Browse to where you copied the Certificate file.

In the File Name text box, hit " Browse " and select the certificate. Click " Next > " . Example of the box filled after cert has been selected:


Certificate Import Wizard

File to Import

Specify the file you want to import.

File name:

C:\Users\TiminsKY\Documents\ENF\Scripts\PS_TiminsKy_Trusted_R

Note: More than one certificate can be stored in a single file in the following formats:

Personal Information Exchange- PKCS #12 (.PFX,.P12)

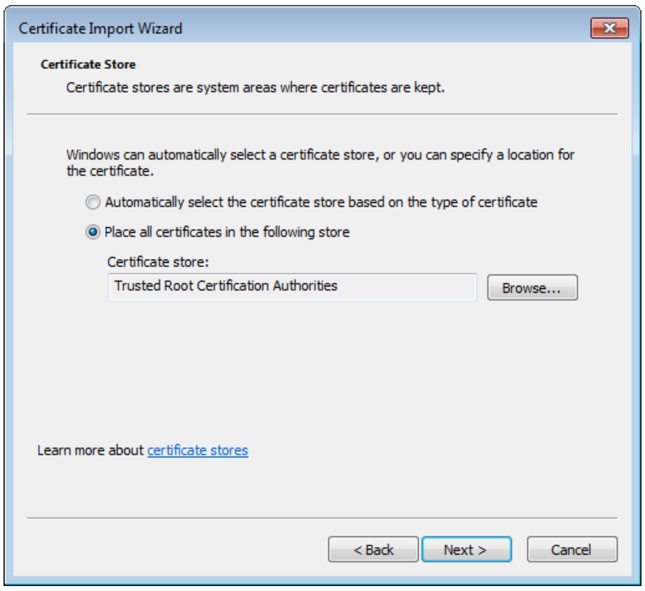
Cryptographic Message Syntax Standard- PKCS #7 Certificates (.P7B)

Microsoft Serialized Certificate Store (.SST)

Learn more about certificate file formats

Figure 10: Figure 2-5 - Select Certificate

1) Click the radio button for " Place all certificates in the following store " and select " Trusted Root Certification Authorities " in the Certificate Store.


Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

( ) Automatically select the certificate store based on the type of certificate

(●) Place all certificates in the following store

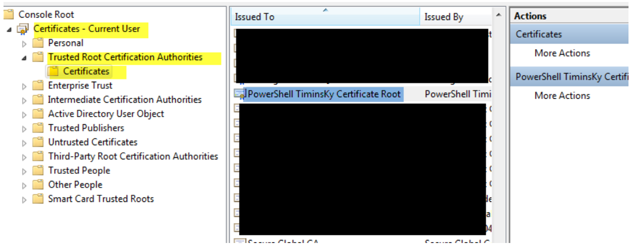
Certificate store:

Trusted Root Certification Authorities

Figure 11: Figure 2-6 - Place in Certificate Store

1) Hit ” Next > ” then hit ” Finish ” .

On the Main MMC screen, you should see ” PowerShell TiminsKy Certificate Root ” in the list of certificates.


Console Root
Certificates - Current User
Personal
Trusted Root Certification Authorities
Certificates
Enterprise Trust
Intermediate Certification Authorities
Active Directory User Object
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authorities
Trusted People
Other People
Smart Card Trusted Roots
Actions
Certificates
More Actions
PowerShell TiminsKy Certifi
More Actions

Issued To	Issued By
[redacted]	[redacted]
PowerShell TiminsKy Certificate Root	PowerShell Timi
[redacted]	[redacted]

Figure 12: Figure 2-7 - Certificate Added to the Certificate Store

1) Click ” File ” -> ” Save ” and save the file. Then exit MMC.

You should now be able to run my scripts.

Running Powershell Scripts

Running Scripts

To run a script, you type ” . \ XXXX.ps1 ” (where XXXX is the name of the script) in the Powershell window (in the directory of the script) and hit enter.

Getting Help for a Script

To get help text for a script, type ” Get-Help . \ XXXX.ps1 –detailed ” (where XXXX is the name of the script).

Running Get-ENFAlerts.ps1

From Powershell If you are using a file called enfrc.json and envrc.json in the same directory as the Get-ENFAlerts.ps1 file, you can just type ” . \ Get-ENFAlerts.ps1 ” into the Powershell window and press enter. If you are using a diffenently named configuration file, add the parameter ” -configFile XXXX ” where XXXX is the name of the configuration file. The same for the environmental configuration file with the parameter ” -envConfigFile ”

For more information on the parameters and the possible ways to use them, type ” Get-Help . \ Get-ENFAlerts.ps1 -Detailed ” into the Powershell window and press enter.

From Batch Script The Get-ENFAlerts.ps1 script can be run from a batch script. It works primarily the same as running from a powershell console, but gives the convenience of double clicking to run. Here is what you would put in the batch script:

```
cd ” C: \ Directory \ where \ the \ Get-ENFAlerts.ps1 \ script \ resides ”
Powershell . \ Get-ENFAlerts.ps1 –configFile . \ enfrc.json
```

How to Run Get-ENFAlerts.ps1 via Scheduled Tasks

Setting up a scheduled task to run the Get-ENFAlerts.ps1 script is not a difficult task. The only difference between how the script is run is that the Starting directory field is used. Here are the steps to creating a scheduled task to run the script.

1) Open the Start Menu and type " Task Scheduler " into the box. At the top of the output, you should see the link to open the scheduler.

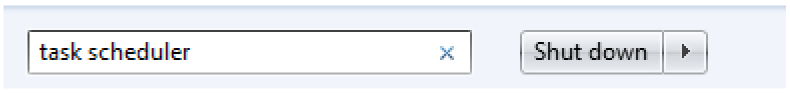


Figure 13: Figure 8 - Start Menu Search

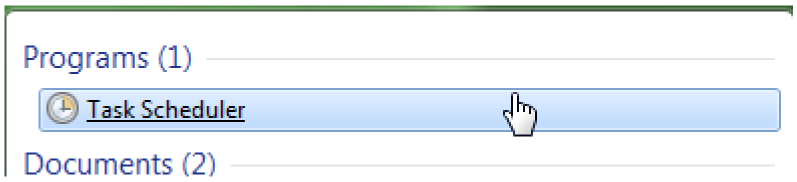


Figure 14: Figure 9 - Start Menu Results

1) On the left hand side of the window, you should see the folder structure of the tasks. Open " Task Scheduler Library " and you should see a folder called " User " . Click this folder. This is where the scheduled tasks will be created.

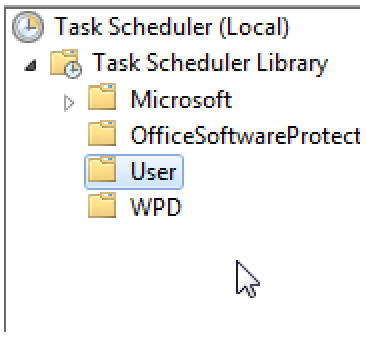


Figure 15: Figure 10 – Task Scheduler Library

1) Right click on the User folder and select ” Create Basic Task ” . (This guide will show how to use the Basic Wizard interface to create a task. If desired, you can simply select ” Create Task ” . This will give you more control over the task itself, but is less intuitive in how to create a task.)

2) In the name field, type in a name that will tell you what the task is. An example would be ” Daily Builds – ENF Alerts – Daily Run. ”

3) In the Description field, you can add more detail so it is more obvious what the task is doing and why it is there. This field is optional.

4) Hit the Next button. You will now be on the ” Trigger ” page. This is how often you want to run the script. (The choices are simplified on this page since it is in the Basic Task Wizard. You will have the option to modify this setting with more granularity after it is created.) After selecting the trigger, hit the Next button.

5) If prompted, select the starting time, date, day of the week, month, etc that is shown on this page. This is only if you selected to have the task work on a timed schedule rather than based on a computer event.

6) On the Action page, select ” Start a Program ” . Hit the Next button.

7) If using pure Powershell to run the script:

   a) In the ” Program/script ” text box, put ” Powershell.exe ” .

   b) In the ” Add Arguments ” section, you are going to add the information on the script itself.

      (i) Here is an example of what it will look like if you use the standard enfrc and envrc (in the same directory as the script) and that directory is ” C: \ ENF \ Scripts ” .

C: \ ENF \ Scripts \ Get-ENFAlerts.ps1 ” enfrc.json ” ” C: \ ENF \ Scripts ”

1) a) (i) Here is an example if you are using custom enfrc configuration file in a sub-folder called ” Config ” and the starting directory is ” C: \ Users \ Timinsky \ scripts \ EnfEmergencyAlerts ”

C: \ Users \ TiminsKy \ Scripts \ EnfEmergencyAlerts \ Get-ENFAlerts.ps1 ” Configs \ DailyBuilds-CL-Property-7DayRollup-enfrc.json ” ” C: \ Users \ TiminsKy \ Scripts \ EnfEmergencyAlerts ”

1) If using the Batch script method:

   a) In the " Program/script " text box, put:

" C: \ Location \ of \ the \ batch \ script \ FILE.bat "

1) Click next and then click save.
2) You have created a scheduled task that will run the script at the desired time/frequency.

It should be noted that when the script runs, it will pop up a PowerShell window for a couple of seconds and then vanish. This is the way that it is designed to work.

How to Remediate an ENF

Determine the Severity of the ENF

While there is a field on the ENF for severity, it is based upon the number of occurrences and is not always a good measure of what the severity actually is. The severity should be determined by the person doing the triage based upon how impactful the error is, how high the count is, and will it impact the end user. These are the factors that should be used if a CSR is created for the ENF.

Adding Comments to the ENF

Comments on a ENF go in the " Comments " textbox in the " Notification Comments " section at the bottom of the main page. Use Ctrl+Enter to create a new line within the text box. Comments should be presented in a newest to latest order, so any new comments should be added to the top of the comments list. Each comment should be in the following format:

[FL YYYY-MM-DD] This is a comment.

Where " FL " is the initials of first and last name, " YYYY " is the year in 4 character format, " MM " is the month in two character format, and " DD " is the day in two character format. An example would be:

[KT 2016-08-01] This is an example comment.

Setting the ENF to In Progress

If a CSR is being created for the ENF, then the ENF should be moved to a status of " In Progress " . It should only be in this state if there is a CSR created for it. If no CSR has been created, then keep the ENF in Not Triaged or ReOpened.

Creating a CSR for the ENF

If it is determined that remediation is needed for the ENF, the next step is to create a CSR for the ENF.

When creating a CSR for an ENF, a few rules should be used to ensure there is the proper amount of information.

- Client: This should be set to " INT " . Do not set it to the client as they should not be able to see the CSRs we create for ENFs.
- Sub Client: If the error impacts a specific client only, then the client should be added to the Sub Client section. At the moment, there are only a hand full of valid clients in this section, so only add if that client exists in the list.
- Title: The title should start with " ENF – " to show that the CSR was created due to an internal ENF.
- Detailed Description: On top of the normal information that should be included in the description, ensure that the GUID for the ENF is included in the following format. This is to allow an easy lookup of the ENF in the tool from the CSR.
  - GUID { XXXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX }
- Detailed Description: It is recommended that the stack trace should be included in the description. The stack trace can be obtained via either the ENF tool, the ENF Alert email, or the email.html found in the ENF zip.
- In the Category field, fill the field with " DBENF " . This is to allow for easy isolation of the Daily Builds ENFs when creating a search in PPM.
- Ensure that that the latest ENF zip from the ENF is attached to the CSR. If possible, more than one is preferable. The link for the latest ENF zip can be found in the details section of the ENF or in the ENF tool.

- It is recommended to attach a polprint of the latest policy on the ENF. This is due to the fact that Daily Builds ENFs do not contain polprints as a way to conserve on drive space. Adding a polprint of the policy as soon after the ENF is created is desired since the policy can be changed after the ENF is created and can make triage & analysis impossible.

Setting the ENF to Resolved

If it has been determined that the ENF is a non-issue, then the status should be set to Resolved. Setting an ENF to Resolved should only be done if there is no remediation for the ENF.

As well, any non ” SS ” ENF should auto resolve from In Progress, Not Triaged, and Re-Opened after 15 days with no occurrences.

ENF Crash Course

ENF Overview – Enterprise Notification Facility

ENF stands for *Enterprise Notification Facility*. On the server side, it is an enterprise level service that standardizes and enhances how application failures are handled. On the user (person triaging) side, it is a collection of information about a crash that occurred within the system.

When an error or crash occurs and there is code in the section to create an ENF, the service creates a ” package ” called an ENF. The package is a zip file that contains information such as stack traces, information overviews, polprints, configuration files, etc. It does a freeze of all these files and contains them in the zip file. Not all information in the zip file will be relevant or useful to the current error at hand. As well, the information is added to a database which stores the information for later retrieval and for use by the ENF utility. (For more information on the database, please see the How to Use the ENF Database section.)

In this document, the terms ENF and Triage ENF will be used. And ENF is a collection of information about a specific "crash" that happened. They all have different GUIDS as their primary key. A Triage ENF is the ENF with the first record of a specific Stack trace. All of the specific crash ENFs are assigned these Triage ENF GUIDS and are referred to as occurrences of a Triage ENF. It is a one-to-many relationship. In most cases, only the Triage ENF and its GUID will be what is discussed and the individual ENFs will only be useful for their SANs, Timestamps, and zip files.

ENF Utility

The ENF utility is used as a front-end to the ENF database that is built in Microsoft Access. From this tool, you can: view detailed information about an ENF, modify information about an ENF, and view reports of top count ENFs for a certain timeframe.

There are three major screen types on the ENF utility: The Unique Notifications Edit screen, the "Reports" style screen, and the "List" style screen. The report and list style screen have multiple variations. For the purposes of this document, the only reports style screens will be TriageTopXAndSSNotifications & GetTopXAndSSNotifications and the only list style screens will be the GetAllDistinctByStack & GetAllDistinctBySAN.

Unique Notifications Edits

The Unique Notifications Edits screen is the main landing screen. It is used to select your connection, select queries, search, and to display information on an ENF. Here is a breakdown of the screen.

Top Bar:

- Connection

- These are the different environments that you can look at. Most are self-explanatory. The only main note is that AllState _ PreProd looks at both AllState QA Environments (PreProd and Test). And DailyBuilds looks at DailyBuilds, the MP0X environments, and the perimeter environments.

After selecting, hit the button to the right of the dropdown.

ENF Filter

- These are basically a set of parameters that are predefined. For instance, you can select the "Billing Team" filter and it will only show ENFs that have their Portfolio as "Billing".

Queries

- These are basically different SQL queries that you can run. Most are self-explanatory.
- GetAllDistinctByXXXXXXX will get all of the ENFs in the current environment that matches the value in the popup. The result will be on another tab and will display in a spreadsheet format. These search every ENF collected in the database (assuming they have not been deleted).

GetTop5AndSSNotifications will get a list of " Top " ENFs (basically the largest counts) and a list of SS ENFs (Ones that have certain types that are considered more important.) They are grouped by ExceptionDate and ordered by TodaysCount. The popups for this are:

- Select a day to look at

How many days to look at. These look at the Month day, so 1 day would include 2015-06-03 and 2015-06-02.

How many top ENFs. We usually look at 20.

TriageTopXAndSSNotifications will get a list similar to the GetTop5AndSSNotificiations, but is defaulted to one day and 20 top. As well, it has a mini selection at the top to view and edit select information. It is more convenient than the GetTop5AndSSNotifications for triaging, but loses some information that is important.

Hit the button on the right of the dropdown to go to the selected screen.

Srch Field & Srch Txt

- This will search the current environment. You select what you want to search by. Most are pretty much self-explanatory. Here are the ones I use the most.
- GUID is the triage GUID.

- When getting lists from the queries, you would use the Triage GUID with this search to see that ENF.

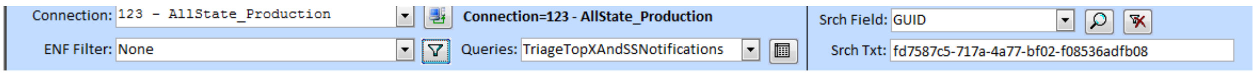
CSR will search all ENFs for ones that contain that CSR.

StackTrace will search for the part of the StackTrace you paste into the txt field.

Exception message is the same as StackTrace but will the Exception Message

SAN will search all of the Triage ENFs' SANs. This is limited to the first SAN reported with the Stack Trace of the Triage ENF. It is better to use GetAllDistinctBySAN as that will search all ENF records instead of only the Triage ENFs.

Press the magnifying glass button on the right to search.



Connection: 123 - AllState_Production
ENF Filter: None
Connection=123 - AllState_Production
Queries: TriageTopXAndSSNotifications
Srch Field: GUID
Srch Txt: fd7587c5-717a-4a77-bf02-f08536adfb08

Figure 16: Figure - The Top Bar of the Main Screen

Main Screen

- Environment is the environment that they were created in.

Occurrences is the count of occurrences since the LAST Reopen (or open if no reopens.)

Reopens is the number of times the Triage ENF has had a status of Re-Open.

Total Occ is the total number of occurrences since the very first open.

Created Date is the timestamp of the first occurrence of that last Reopen.

Last Date is the timestamp of the last occurrence to happen.

User Email, SAN, PolicyNo, and User Name are from the first occurrence of the enf.

Status

- Not triaged: Nothing with the ENF has been done

In Progress: Something is happening with it. A CSR is created. I've been using that for when I send emails to people about them.

Resolved: Marking that something is complete. DON'T actually use this. If you set to resolved before the code has shipped back, it will just reopen the ENF.

Re-Opened: Same as Not Triaged, just that the ENF has had an occurrence since it was resolved or auto-resolved.

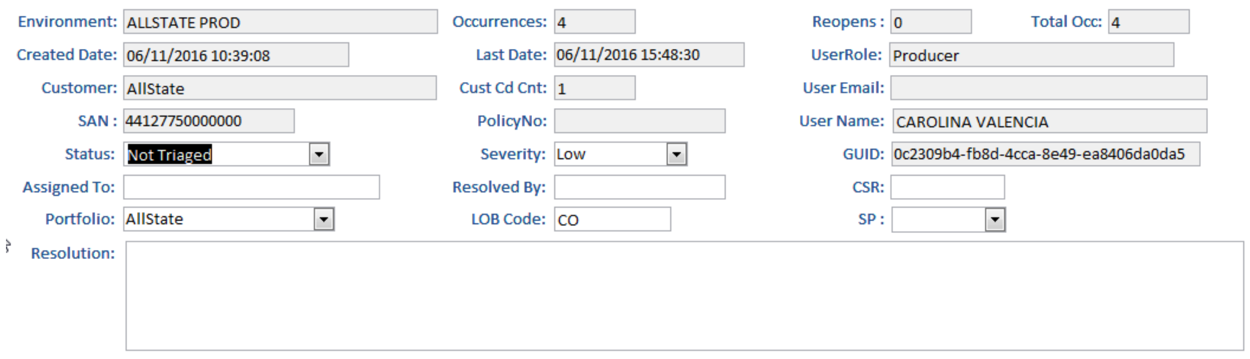
Auto-Resolved. The system sets the ENF to Auto-resolved after a period of time of no occurrences in 30 days. In Allstates environment, this only happens to TOP ENFs, not to SS ENFs.

Severity: Determined by number of occurrences and type.

GUID: The Triage GUID of the ENF collection.

Assigned To and CSR: These you manually enter to store info on the who is working on it and the CSR.

LOB Code: This is what comes through the ENF as the LOB causing the issue. It is usually correct.



Environment:	ALLSTATE PROD	Occurrences:	4	Reopens :	0	Total Occ:	4
Created Date:	06/11/2016 10:39:08	Last Date:	06/11/2016 15:48:30	UserRole:	Producer		
Customer:	AllState	Cust Cd Cnt:	1	User Email:			
SAN :	44127750000000	PolicyNo:		User Name:	CAROLINA VALENCIA		
Status:	Not Triaged	Severity:	Low	GUID:	0c2309b4-fb8d-4cca-8e49-ea8406da0da5		
Assigned To:		Resolved By:		CSR:			
Portfolio:	AllState	LOB Code:	CO	SP :			
Resolution:							

Figure 17: Figure - Main Section of the Main Screen

Original Notification Error Info

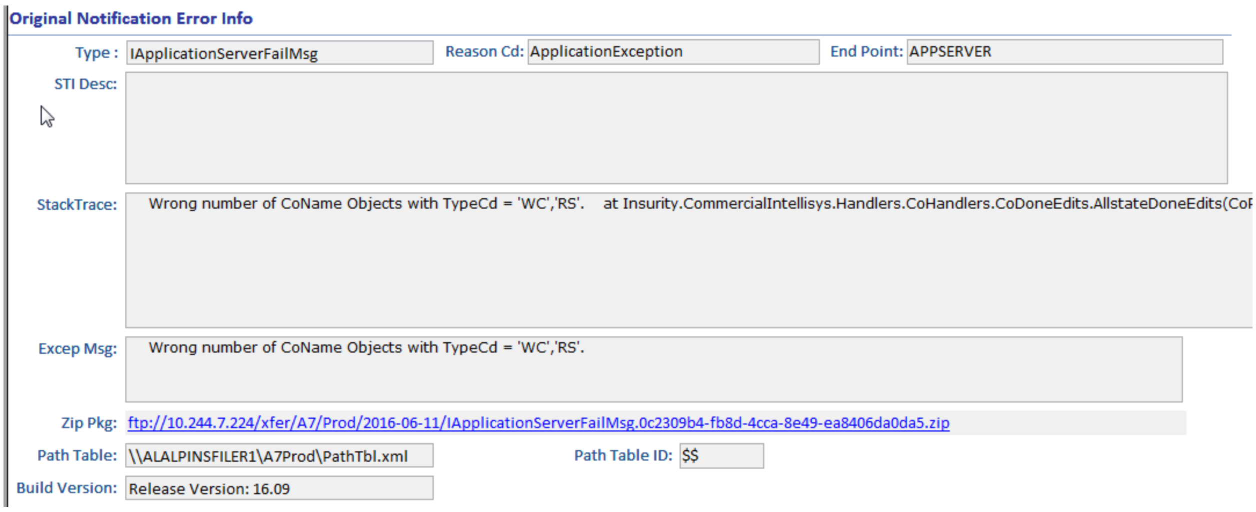
- Type is the type of ENF thrown as defined in the code and ServiceBus.

This is the First occurrence information

- StackTrace: This is the stack for the error. It is what determines if an occurrence is the same ENF. So all occurrences have the same stack as this section.

Excep Msg: This is the Exception Message. All occurrences have the same exception message as this section.

Zip Pkg: This is the zip for the first occurrence.



Original Notification Error Info

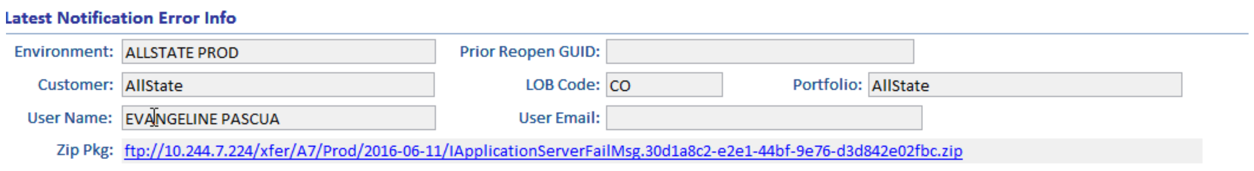
Type :	IApplicationServerFailMsg	Reason Cd:	ApplicationException	End Point:	APPSERVER
STI Desc:					
StackTrace:	Wrong number of CoName Objects with TypeCd = 'WC','RS'. at Insurity.CommercialIntellisys.Handlers.CoHandlers.CoDoneEdits.AllstateDoneEdits(CoF				
Excep Msg:	Wrong number of CoName Objects with TypeCd = 'WC','RS'.				
Zip Pkg:	ftp://10.244.7.224/xfer/A7/Prod/2016-06-11/IApplicationServerFailMsg.0c2309b4-fb8d-4cca-8e49-ea8406da0da5.zip				
Path Table:	\\ALALPINSFILER1\A7Prod\PathTbl.xml			Path Table ID:	$$
Build Version:	Release Version: 16.09				

Figure 18: Figure - Original Notification Information Section of the Main Screen

Latest Notification Error Info

- This is the last occurrence.

Environment: The environment of the ENF.


Latest Notification Error Info

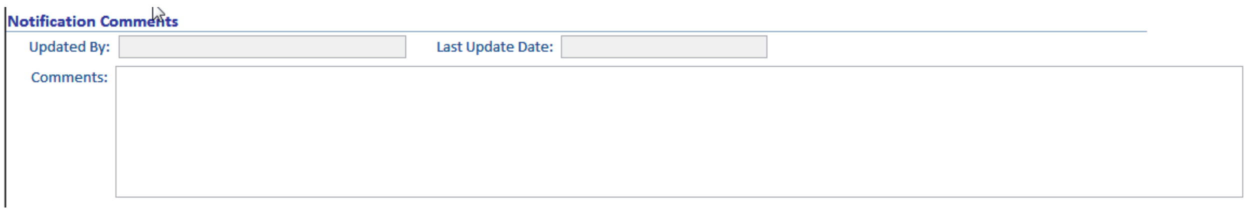
Environment:	ALLSTATE PROD	Prior Reopen GUID:			
Customer:	AllState	LOB Code:	CO	Portfolio:	AllState
User Name:	EVANGELINE PASCUA	User Email:			
Zip Pkg:	ftp://10.244.7.224/xfer/A7/Prod/2016-06-11/IApplicationServerFailMsg.30d1a8c2-e2e1-44bf-9e76-d3d842e02fbc.zip				

Figure 19: Figure - Latest Notification Information Section of the Main Screen

Notification Comments

- This is where comments about the ENF are put. They are usually prepended with your initials and the date in ISO format (YYYY-MM-DD) all surrounded by square brackets followed by the comment. Use Ctrl+Enter to make a new line.

To save any edits made, you must click the dark grey bar on the left side of the screen. It will get darker after it has been click signaling the save happened.


Notification Comments

Updated By:

Last Update Date:

Comments:

Figure 20: Figure - Comments Section of the Main Screen

Queries

Note: When moving around, I would use arrow keys. If you click, just press down arrow then up arrow to highlight the field to copy.

GetAllDistinctByStack: Showing the useful columns. The rest should be self-explanatory. (Should cover all the GetAllDistinctByXXXXXX.)

- Triage GUID is the GUID you would use to search for the ENF.

GUID is distinct for each occurrence.

MessageCreateDate is the timestamp of the occurrence.

AlertZipFilePath is the location of the zip on the FTP server. Just paste the URL into a browser to download.

GetTop5AndSSNotifications: Showing useful columns. The rest should be self-explanatory.

- CAT: TOP is count based, SS are showstoppers despite count.

TodaysCount: The current count for how many days you entered. Remember 1 day means 2015-06-03 and 2015-06-02.

TriageGUID: What you would search the ENF by.

GetTopXAndSSNotifications: This is similar to the GetTop5AndSSNotifications, but contains some of the information about the selected ENF above the speadsheet. The only editable fields on this screen are CSR, Status, and Severity. As well, if you click the magnifying glass on the right of the TriageGUID field, it will bring you to the Unique Notifications Edit screen with the ENF loaded to the one that was on the other screen.

How to edit an ENF to add information

Open an ENF

Open the ENF utility. On the Unique Notification page, drop down the Connection drop-down and select the connection to the DB you need. Click the button next to the dropdown. In the Srch Field, drop down the dropdown and select "GUID". Paste the GUID into the Srch Txt field. Then click the magnifying glass button. Now the ENF is loaded on the page.

Add a CSR to the ENF

In the top section (of the white part of the page), the sixth field down on the right column is the CSR section. Enter/paste the CSR number in that box. Then, the fifth field down on the left column is the Status field. Select "In Progress" from the dropdown. Then click the dark grey vertical bar on the left hand side to save the changes.

How to add comments to an ENF

Go to the bottom of the Unique Notifications page and you will see a large text box labeled Comments. Enter any comments needed into this box. Prepend any new comments with the text "[XX YYYY-MM-DD]" where XX is your initials. If you need a new line, press Ctrl+Enter for a new line. After you have added your comments, click the dark grey bar on the left to save your comments.

How to Use the ENF Database

The ENF database can be a good way to monitor ENFs and to get information with specific modifiers. It is recommended that you only use the database to view information and not for modifying the ENF.

Connect to the ENF Database

In SQL Server Management Studio, connect to the ENF database using the following information.

Server: "HFDWPSQLV4 \ DailyBuilds02"
Authentication: "SQL Server Authentication"
User: "GARDEF"
Password: "GARDEF"

Databases

Daily Builds: "DailyBuild _ Alerts _ Test"
Allstate Prod: "Production _ Alerts _ A7"
Allstate QA ENVs: "Production _ Alerts _ A7 _ PreProd"

Tables

Triage ENFs: "Unique _ Alerts"
Occurrence ENFs : "Distinct _ Alerts"

Basic SQL Statetments

Get the Triage information for a Triage GUID.

```
SELECT * FROM [Unique _ Alerts] AS UA
  WHERE [UA].[GUID] = 'xxxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx'
```

Get all the occurrences of a Triage GUID/ENF.

```
SELECT * FROM [Distinct _ Alerts] AS DA
  WHERE [DA].[StackTrace] =
    (SELECT [UA].[StackTrace] FROM [Unique _ Alerts] as UA
     WHERE [UA].[GUID] = 'xxxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx')
```

Get all occurrences of ENFs within the Property LOBs

```
SELECT [DA].[MessageCreatedDate],*
FROM [Distinct _ Alerts] AS DA
  WHERE [DA].[Servicelobcd] IN ('PF','IM','IA','IB','OM','OA','OP','BM')
    AND [DA].[Portfolio] = 'CL'
    AND [DA].[MessageCreatedDate] > '07/10/2016'
  ORDER BY [DA].[MessageCreatedDate] DESC
```