Zabbix 3.0 の新機能

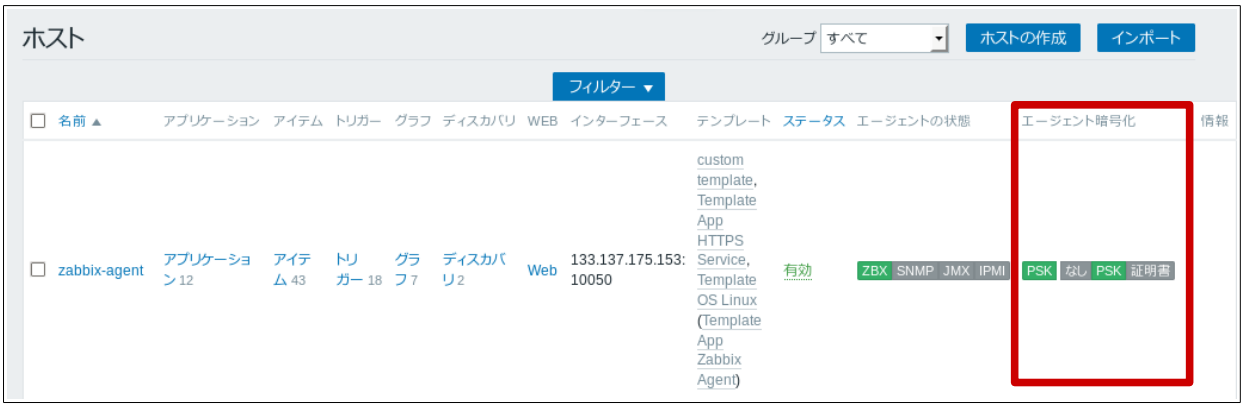
通信暗号化と認証

Zabbix サーバ、プロキシ、エージェント、zabbix_sender および zabbix_get 間で TLS 1.2 を用いた暗号化通信が可能になりました。認証の方法は、コンポーネントごとに証明書もしくは PSK (Pre-Shared Key) から選択します。Zabbix デーモンが暗号化通信に用いるポートは暗号化されていない通信と同じポートを用いるため、別途ファイアウォールの設定などを行なう必要はありません。

Web インターフェースの変更

暗号化通信機能の追加に伴い、Web インターフェースにも暗号化関連の項目が追加されました。

- [設定] - [ホスト] 画面のエージェント暗号化カラム



- ホスト設定画面の [暗号化] タブ

- [管理] - [プロキシ] 画面の暗号化カラム

- プロキシ設定画面の [暗号化] タブ

暗号化に使用されるライブラリ

暗号化通信に使用できるライブラリは以下の 3 つです。

- mbed TLS 1.3.9 以降 (2.x はサポートされていません)
- GnuTLS 3.1.18 以降
- OpenSSL 1.0.1 以降

ソースからコンパイルする際に暗号化機能を有効にするには、どのライブラリを用いるかをオプションで明示的に指定する必要があります。

- `--with-mbedtls[=DIR]`
- `--with-gnutls[=DIR]`
- `--with-openssl[=DIR]`

```
$ ./configure --enable-server --enable-agent --with-mysql \
--with-openssl --with-net-snmp --with-libcurl --with-libxml2
```

Zabbix 社が配布している rpm パッケージでは、OpenSSL を使用するようになっています。

各コンポーネントで暗号化通信が使用可能かどうかは、起動後のログで確認可能です。

PSK による暗号化通信

PSK による暗号化通信では、各コンポーネント間で PSK と PSK アイデンティティの組を用いて認証を行ないます。

◯ PSK の生成

PSK は 16 進数の文字列である必要があります。また、Zabbix で許容される PSK のサイズは 16byte 以上 256byte 以下です。例えば、OpenSSL を用いた以下のコマンドにより、32byte の 16 進数文字列を生成することができます。

```
$ openssl rand -hex 32
67913c6ef927b10645897867bdc0517b3f1106266e94054d1dc5135d852d1c3e
```

◯ サーバ - エージェント間の PSK による暗号化通信設定

1. エージェント側のホストに PSK をファイルとして設置します。

```
$ openssl rand -hex 32 > /etc/zabbix/zabbix_agentd.psk
$ chown zabbix.zabbix /etc/zabbix/zabbix_agentd.psk
$ chmod 600 /etc/zabbix/zabbix_agentd.psk
```

2. Zabbix エージェントの設定ファイル (zabbix_agentd.conf) の TLS パラメータを変更します。

パラメータ	備考
TLSConnect=psk	エージェントがアクティブチェックでサーバやプロキシに接続する手法
TLSAccept=psk	エージェントへの接続手法
TLSPSKFile=/etc/zabbix/zabbix_agentd.psk	PSK ファイルのフルパス
TLSPSKIdentity=PSK ID 001	PSK を識別するために用いられる任意の文字列 システム内で一意である必要がある

3. Web インターフェースの [設定] - [ホスト] で対象ホストを選択し、[暗号化] タブから暗号化通信設定を行ないます。

   ホストへの接続およびホストからの接続として「PSK」を選択し、PSK アイデンティティの欄にエージェントの設定ファイルで設定した TLSPSKIdentity と同じ文字列を、PSK の欄に PSK の値をそれぞれ入力します。

◯ プロキシに対する PSK を用いた暗号化通信設定

プロキシに対する PSK を用いた暗号化通信設定は、サーバ – エージェント間の設定と同様に行ないます。

証明書による暗号化通信

証明書による暗号化通信では、各コンポーネント間で CA により署名された PEM 形式の RSA 証明書を用いて認証を行ないます。自己署名証明書はサポートされません。

◯ サーバ - エージェント間の証明書による暗号化通信設定

1. サーバ側のホストに CA 証明書 (zabbix_ca_file)、サーバ証明書 (zabbix_server.crt)、サーバの秘密鍵 (zabbix_server.key) を配置します。

2. Zabbix サーバの設定ファイル (zabbix_server.conf) の TLS パラメータを変更します。

パラメータ	備考
`TLSCAFile=` `/etc/zabbix/zabbix_ca_file`	最上位 CA 証明書ファイルのフルパス 証明書チェーンの場合は下位から上位の順で、複数の CA からの証明書はその全てをこのファイルに含める
`TLSCertFile=` `/etc/zabbix/zabbix_server.crt`	サーバ証明書ファイルのフルパス 証明書チェーンの場合はサーバ証明書を最初に、続けて下位から上位の順で CA 証明書をこのファイルに含める
`TLSKeyFile=` `/etc/zabbix/zabbix_server.key`	サーバの秘密鍵のフルパス

3. エージェント側のホストに CA 証明書 (zabbix_ca_file)、エージェント証明書 (zabbix_agentd.crt)、エージェントの秘密鍵 (zabbix_agentd.key) を配置します。

4. Zabbix エージェントの設定ファイル (zabbix_agentd.conf) の TLS パラメータを変更します。

   TLSServerCertIssuer および TLSServerCertSubject は必須項目ではありませんが、設定することによりエージェントのセキュリティを高めることができます。

パラメータ	備考
`TLSConnect=cert`	エージェントがアクティブチェックでサーバやプロキシに接続する手法
`TLSAccept=cert`	エージェントへの接続手法
`TLSCAFile=` `/etc/zabbix/zabbix_ca_file`	最上位 CA 証明書ファイルのフルパス 証明書チェーンの場合は下位から上位の順で、複数の CA からの証明書はその全てをこのファイルに含める

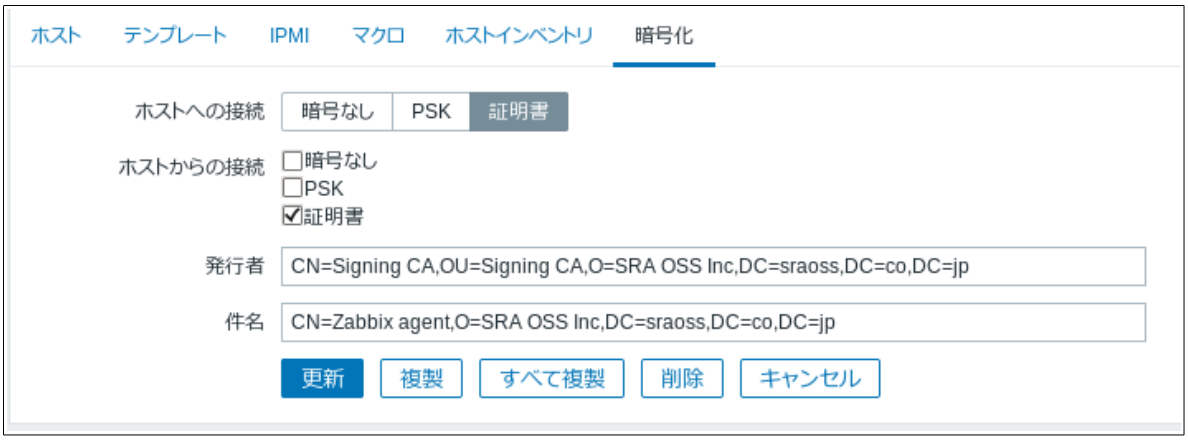
`TLSServerCertIssuer=`
`CN=Signing CA,OU=Signing CA,`
`O=SRA OSS Inc,`
`DC=sraoss,DC=co,DC=jp`
	サーバ証明書の発行者

`TLSServerCertSubject=`
`CN=Zabbix server,`
`O=SRA OSS Inc,`
`DC=sraoss,DC=co,DC=jp`
	サーバ証明書の件名

パラメータ	備考
TLSCertFile= /etc/zabbix/zabbix_agentd.crt	エージェント証明書ファイルのフルパス 証明書チェーンの場合はエージェント証明書を最初に、続けて下位から上位の順で CA 証明書をこのファイルに含める
TLSKeyFile= /etc/zabbix/zabbix_agentd.key	エージェントの秘密鍵のフルパス

5. Web インターフェースの [設定] - [ホスト] で対象ホストを選択し、[暗号化] タブから暗号化通信設定を行ないます。

   ホストへの接続およびホストからの接続として「証明書」を選択し、発行者の欄にエージェント証明書の発行者を、件名の欄にエージェント証明書の件名をそれぞれ入力します。こちらの発行者および件名もセキュリティを高めるためには設定したほうがよいでしょう。



◯ プロキシに対する証明書を用いた暗号化通信設定

プロキシに対する証明書を用いた暗号化通信設定も、サーバ - エージェント間の設定と同様に行ないますが、アクティブモードの場合には Zabbix プロキシの設定ファイル (zabbix_proxy.conf) の TLSConnect を cert、Web インターフェースの [管理] - [プロキシ] における暗号化タブのプロキシからの接続を証明書に設定します。

パッシブモードの場合には Zabbix プロキシの設定ファイルの TLSAccept を cert、Web インターフェースの [管理] - [プロキシ] における暗号化タブのプロキシへの接続を「証明書」、プロキシからの接続を「暗号なし」に設定します。

zabbix_get および zabbix_sender を用いた暗号化通信

zabbix_get および zabbix_sender も PSK および証明書を用いた暗号化通信をサポートするようになりました。

◯ zabbix_get の暗号化通信

暗号化通信機能の追加に伴い、zabbix_get に以下のオプションが追加されました。

オプション	引数
--tls-connect *value*	エージェントへの接続法 value: uncrypted、psk、cert
--tls-ca-file *CA-file*	CA 証明書のフルパス
--tls-crl-file *CRL-file*	CRL ファイルのフルパス
--tls-agent-cert-issuer *cert-issuer*	エージェント証明書の発行者
--tls-agent-cert-subject *cert-subject*	エージェント証明書の件名
--tls-cert-file *cert-file*	証明書ファイルのフルパス
--tls-key-file *key-file*	秘密鍵のフルパス
--tls-psk-identity *PSK-identity*	PSK アイデンティティの文字列
--tls-psk-file *PSK-file*	PSK ファイルのフルパス

以下は PSK および証明書を用いた zabbix_get の実行例です。

```
$ zabbix_get -s 133.137.175.153 -k "agent.ping" \
--tls-connect psk \
--tls-psk-identity "PSK ID 001" \
--tls-psk-file /etc/zabbix/zabbix_agentd.psk

1
```

```
$ zabbix_get -s 133.137.175.180 -k "agent.ping" \
--tls-connect cert \
--tls-ca-file /etc/zabbix/zabbix_ca_file \
--tls-agent-cert-issuer \
"CN=Signing CA,OU=Signing CA,O=SRA OSS  Inc,DC=sraoss,DC=co,DC=jp"\
--tls-agent-cert-subject \
"CN=Zabbix agent,O=SRA OSS Inc,DC=sraoss,DC=co,DC=jp" \
--tls-cert-file /etc/zabbix/zabbix_server.crt \
--tls-key-file /etc/zabbix/zabbix_server.key

1
```

◯ zabbix_sender の暗号化通信

暗号化通信機能の追加に伴い、zabbix_sender に以下のオプションが追加されました。

オプション	引数
`--tls-connect` *value*	エージェントへの接続法 value: uncrypted、psk、cert
`--tls-ca-file` *CA-file*	CA 証明書のフルパス
`--tls-crl-file` *CRL-file*	CRL ファイルのフルパス
`--tls-server-cert-issuer` *cert-issuer*	サーバ証明書の発行者
`--tls-server-cert-subject` *cert-subject*	サーバ証明書の件名
`--tls-cert-file` *cert-file*	証明書ファイルのフルパス
`--tls-key-file` *key-file*	秘密鍵のフルパス
`--tls-psk-identity` *PSK-identity*	PSK アイデンティティの文字列
`--tls-psk-file` *PSK-file*	PSK ファイルのフルパス

以下は PSK および証明書を用いた zabbix_sender の実行例です。

```
$ zabbix_sender -z 133.137.175.152 -s "zabbix-agent" \
-k "trapper" -o 10 \
--tls-connect psk \
--tls-psk-identity "PSK ID 001" \
--tls-psk-file /etc/zabbix/zabbix_agentd.psk

info from server: "processed: 1; failed: 0; total: 1; seconds spent:
0.000094"
sent: 1; skipped: 0; total: 1
```

```
$ zabbix_sender -z 133.137.175.152 -s "zabbix-agent2" \
-k "trapper" -o 10 \
--tls-connect cert \
--tls-server-cert-issuer \
"CN=Signing CA,OU=Signing CA,O=SRA OSS Inc,DC=sraoss,DC=co,DC=jp" \
--tls-server-cert-subject \
"CN=Zabbix server,O=SRA OSS Inc,DC=sraoss,DC=co,DC=jp" \
--tls-ca-file /etc/zabbix/zabbix_ca_file \
--tls-cert-file /etc/zabbix/zabbix_agentd.crt \
--tls-key-file /etc/zabbix/zabbix_agentd.key

info from server: "processed: 1; failed: 0; total: 1; seconds spent:
0.000100"
sent: 1; skipped: 0; total: 1
```

予測検知

監視アイテムの実際の収集値ではなく、その収集値の傾向を分析し、未来の予測値や、特定の値になるまでの予測時間を通知することができるようになりました。これによって、実際に問題が起きる前にその可能性を認識して、予防することができるようになります。

このために、予測関数として `forecast()`、`timeleft()` が追加されました。これは計算アイテムやトリガーの条件として使用することができます。

未来の時点のデータを予測

記法	`forecast (sec|#num,<time_shift>,time,<fit>,<mode>)`
引数	1. `sec` または `#num` 　◦ 何秒間の収集値を見るか、あるいは、最新いくつの収集値を見るかを指定します。 2. `time_shift` 　◦ 指定秒数さかのぼった値を用いる場合に指定します。指定は任意です。 3. `time` 　◦ 予測期間を秒で指定します。 4. `fit` 　◦ 予測方法を指定します。指定は任意です。 　◦ linear: 直線（デフォルト） 　◦ polynomialN: 多項式（数値 N は 1 から 6 まで） 　◦ exponential: 指数関数 　◦ logarithmic: 対数関数 　◦ power: べき乗 5. `mode` 　◦ 出力値を指定します。指定は任意です。 　◦ value: 値そのまま（デフォルト） 　◦ max: 最大値 　◦ min: 最小値 　◦ delta: 最大値 - 最小値 　◦ avg: 平均
備考	• 整数、小数のアイテムに対して使うことができます。 • この関数の返り値の最大値は 999999999999.9999 です。 • 設定が誤っている場合は-1 が返ります。
設定例	• `forecast(#10,,1h)`: 最新 10 この値から予測される、1 時間後の値 • `forecast(1h,,30m)`: 直近 30 分の値から予測される、30 分後の値 • `forecast(1h,1d,12h)`: 直近半日の値から予測される、12 時間後の値

しきい値に達するまでの秒数を予測

記法	timeleft (sec|#num,<time_shift>,threshold,<fit>)
引数	1. sec または #num ○ 何秒間の収集値を見るか、あるいは、最新いくつの収集値を見るかを指定します。 2. time_shift ○ 指定秒数さかのぼった値を用いる場合に指定します。指定は任意です。 3. threshold ○ 通知するしきい値を指定します。 4. fit ○ 予測方法を指定します。指定は任意です。 ○ linear: 直線（デフォルト） ○ polynomialN: 多項式（数値 N は 1 から 6 まで） ○ exponential: 指数関数 ○ logarithmic: 対数関数 ○ power: べき乗
備考	• 整数、小数のアイテムに対して使うことができます。 • この関数の返り値の最大値は 999999999999.9999 です。 • 設定が誤っている場合は-1 が返ります。
設定例	• timeleft(#10,,0): 最新 10 この値から予測される、0 に達するまでの秒数 • timeleft(1h,,100): 直近 1 時間の値から予測される、100 に達するまでの秒数 • timeleft(1h,1d,0): 1 日前の直近 1 時間の値から予測される、0 に達するまでの秒数

使用例

ファイルシステム使用・空き容量の監視で、「/ の空き容量が 0 になるまであと 10 時間と予測したら通知する」ということをします。

1. `timeleft()` を使った計算アイテムを登録します。

タイプ	「計算」を選択します。
キー	任意の名称をつけます。
式	「`timeleft("vfs.fs.size[/,pfree]",1h,,0)`」と入力します。
データ型	「数値（整数）」を選択します。
データの形式	「10 進数」を選択します。
単位	秒を意味する「s」を選択します。

アイテムの作成

[設定] > [ホスト（テンプレート）] > 「アイテム」リンク > 「アイテムの作成」ボタン

2. トリガーを登録します。

条件式	「{ホスト名:timeleft_vfs_size_free.last()}<10h」と入力します。

トリガーの作成

[設定] > [ホスト（テンプレート）] > 「トリガー」リンク > 「トリガーの作成」ボタン

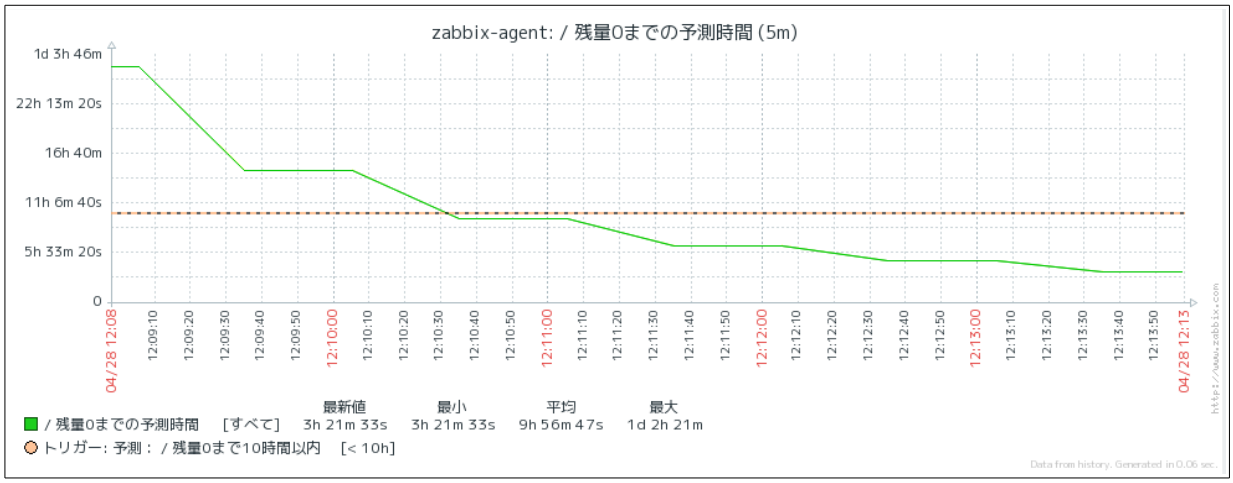
3. ディスクの空き容量を 10%以下まで減らします。

```
# cat add_file.sh
#!/bin/sh

i=0
while :; do
    i=`expr $i + 1`
    dd if=/dev/zero of=/root/test/tempfile_$i bs=1M count=100 >
/dev/null 2>&1
    echo $i `df -h | grep sda | awk '{print $5}'`
    sleep 5
done;

# ./add_file.sh
1 26%
2 27%
3 28%
:
116 88%
117 89%
118 90%
```

4. グラフを確認します。



グラフ

[監視データ] > [最新データ] > 「/残量 0 までの予測時間」の「グラフ」リンク

5. イベントを確認します。

イベント

[監視データ] > [イベント] > イベントのリンク

監視と障害通知

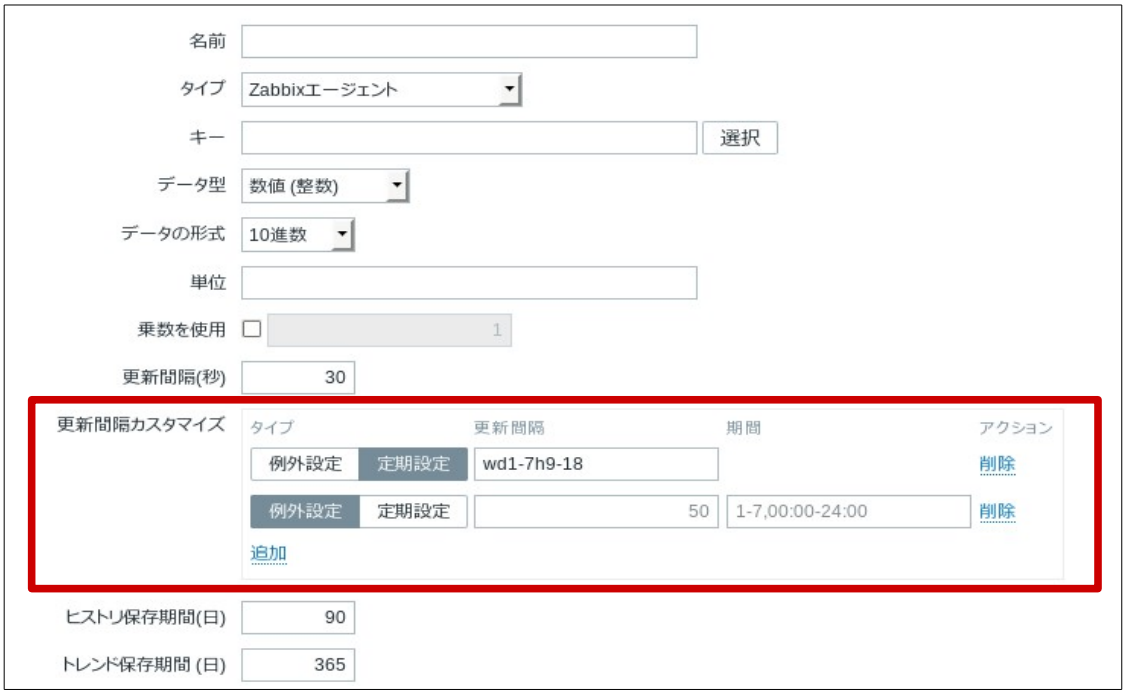
任意のタイミングでアイテムのデータ取得

指定した時刻、時間帯にのみデータを取得することができるようになりました。

これまではZabbixサーバ全体でのメンテナンス期間は定義することはできましたが、アイテムごとにそうした指定はできませんでした。そのため、例えば「夜間はこれらのログには必ずエラーが出力されるので、そのログ監視結果は通知しない」というような要件を満たすには、アイテムは常に収集しながら、トリガーごとにそうした設定を行なう必要がありました。

アイテム作成画面の「更新間隔カスタマイズ」の欄の「定期実行」で、監視タイミングをcronのような表記で設定することができます。

設定例	
h9-17/2	9時～17時の間、2時間おきに実行
wd1-5h9-18	毎週月曜日～金曜日、0時～18時の間、1時間おきに実行
md1wd1h9m30	毎月1日、月曜日であれば9時半に実行
h9m/30	9:00と9:30に実行



アイテムの作成

[設定] > [ホスト（テンプレート）] > 「アイテム」リンク > 「アイテムの作成」ボタン

SMTP 認証の対応

SMTP 認証と SSL/TLS 接続設定に対応しました。また、SMTP サーバのポート番号を指定できるようになりました。

これまでは、SMTP 認証やデフォルトの 25 番ポートでない SMTP サーバを使う場合には、メディアタイプを「メール」ではなく「スクリプト」として、ユーザが独自にスクリプトを用意して使うことで対応していました。

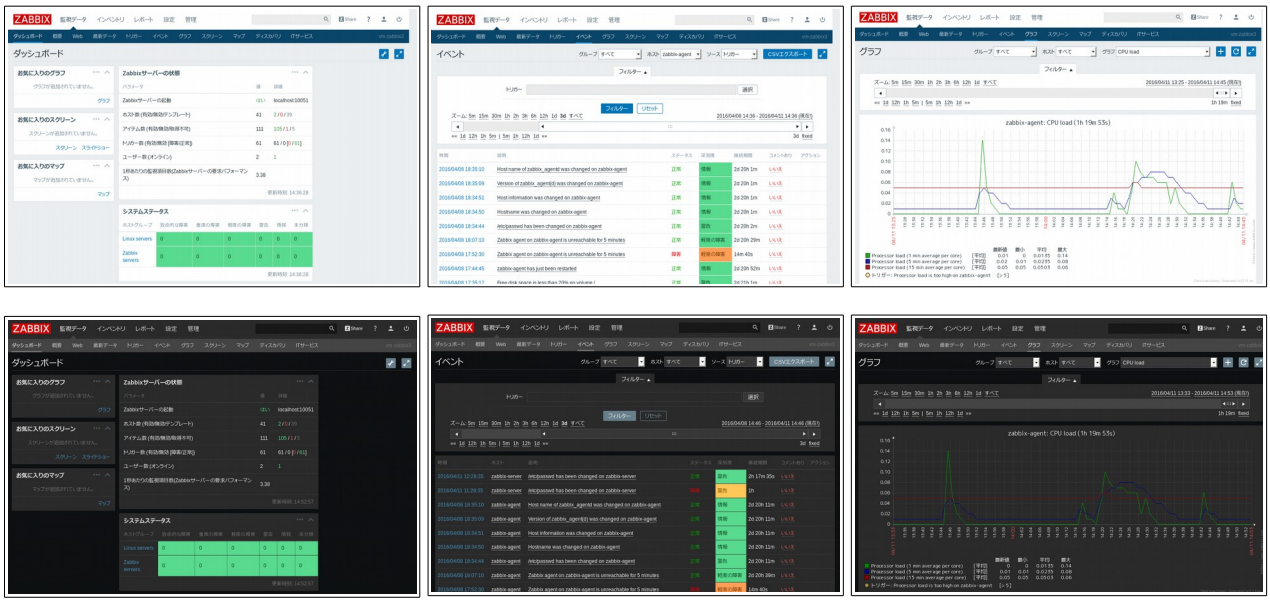
SMTP の設定

[管理] > [メディアタイプ] > 「Email」リンク

Web インターフェース

Web インターフェースの新デザイン

Web インターフェースの PHP が大幅にリファクタリングされました。また、デザインも一新され、従来のデザインを踏襲した Blue テーマの他に Dark テーマも追加されました。



その他に以下のような改善・変更もされています。

- 第１レベルのナビゲーションメニュー（監視データ、インベントリ、レポート、設定、管理）をクリックすると第２レベルのメニューが切り替わるようになりました。これまではクリックだけでなくマウスオーバーでも切り替わっていましたが、好みの分かれるものでした。
- 「マップ」「スクリーン」「スライドショー」は、これまでナビゲーションの「設定」の下にありましたが、「監視データ」の下に変わりました。
- 「ユーザ」と「ユーザグループ」が別のメニューになりました。
- 「監査」「通知レポート」は、これまで「管理」の下にありましたが、「レポート」の下に変わりました。

ユーザごとのマップ、スクリーン、スライドショーと共有

これまでナビゲーションの「設定」の下にありましたが、「監視データ」の下に変わりました。

また、マップ、スクリーン、スライドショーのアクセス権が見直され、一般ユーザでも作成できるようになりました。これまで、マップ、スクリーン、スライドショーを設定することができるのは管理者ユーザだけで、全ユーザで共有のものでした。そのため、一般ユーザが自分しか使わないようなマップ等でも、管理者ユーザに作成を依頼する必要がありました。

Zabbix 3.0 では、そうした設定権限がない一般ユーザーも、マップ、スクリーン、スライドショーを作成できるようになりました。また、作成したものを「非公開」とすれば、ユーザ個人ごとのものになります。「公開」にすると他のユーザーやグループと共有することもできます。

マップの作成

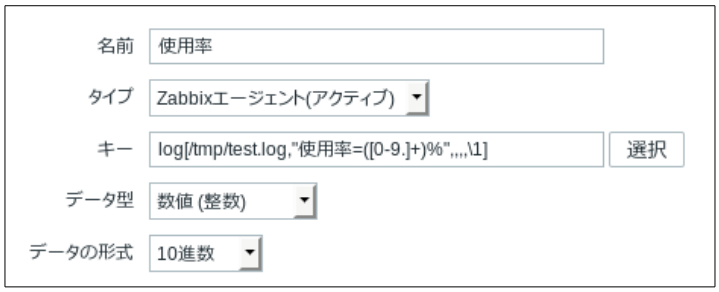
[監視データ] > [マップ] > 「マップの作成」ボタン > 「共有」タブ

ログから数値取得

ログ監視で数値を抜き出して登録するアイテムは、これまでログデータ型としてしか登録できませんでしたが、数値として登録できるようになりました。これによって、値を数値として監視したり、グラフを描画したりできるようになりました。

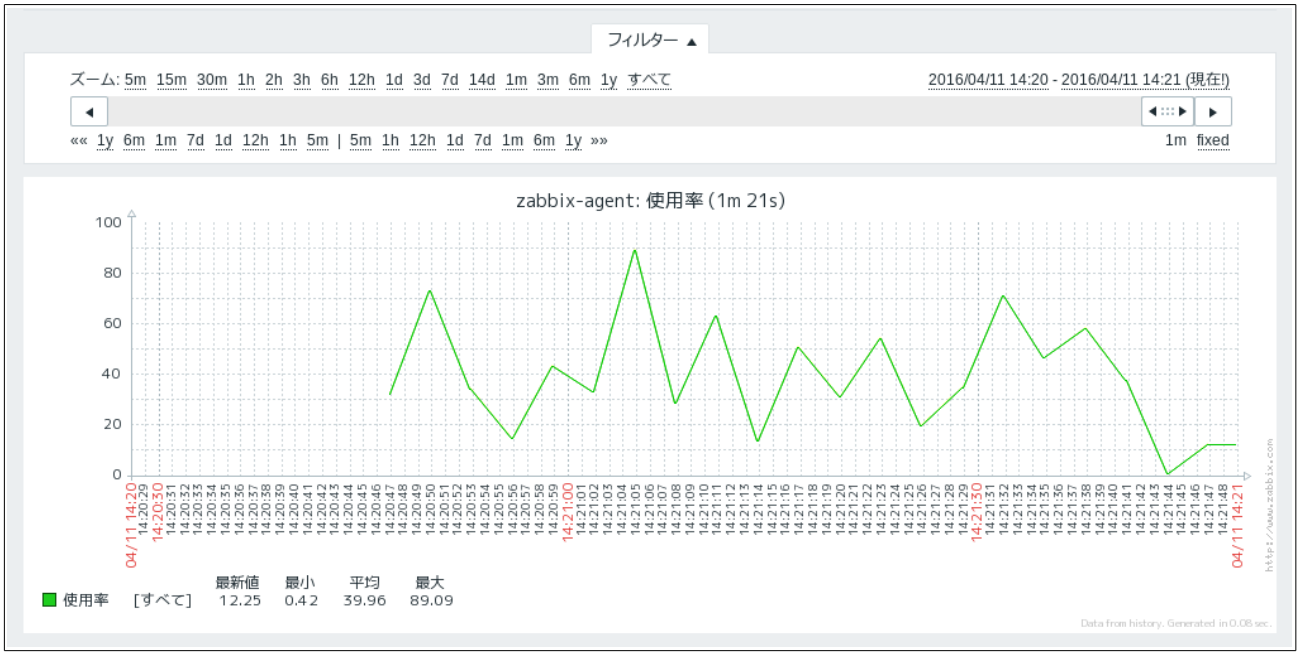
```
$ tail /tmp/test.log
使用率=12.25%
使用率=31.10%
使用率=73.91%
使用率=45.89%
使用率=7.13%
使用率=16.72%
使用率=51.47%
```

```
使用率=9.93%
使用率=65.15%
使用率=54.26%
```



アイテムの作成

[設定] > [ホスト（テンプレート）] > 「アイテム」リンク > 「アイテムの作成」ボタン



アイテム収集値のグラフ

[監視データ] > [最新データ] > 「グラフ」リンク

値のマッピングのエクスポートとインポート

値のマッピングを XML にエクスポートしたり、XML からインポートできるようになりました。

設定のインポート

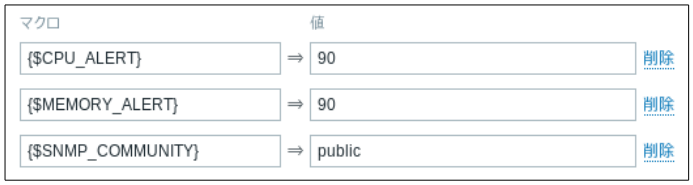
[設定] > [ホスト] または [テンプレート] > 「インポート」ボタン

より詳細なグラフの期間表示

グラフの期間の最小単位が 1 分になりました。

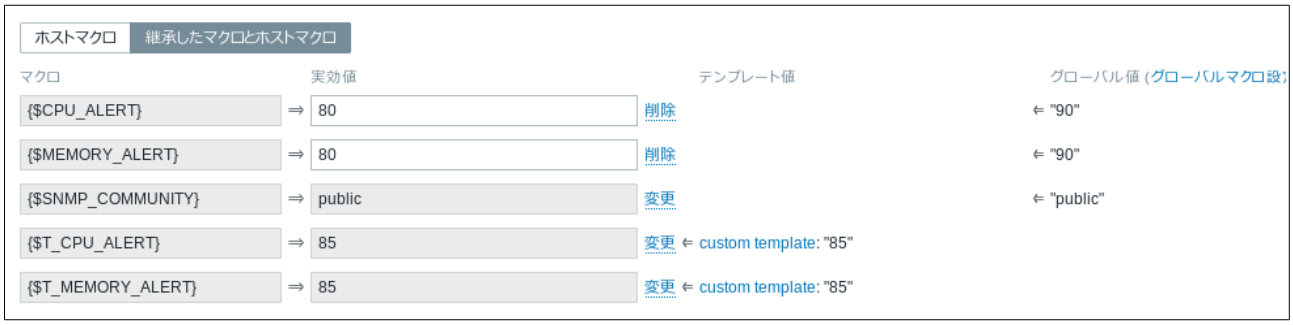
ユーザー定義マクロの表示

ホストやテンプレートのマクロ一覧で、グローバル、テンプレート、ホストのどこで設定されているか、確認できるようになりました。また、実際に利用される値を確認できるようになりました。



グローバルマクロの設定

[管理] > [一般設定] > 「マクロ」選択



ホストマクロの設定

[設定] > [ホスト] > ホスト名 > 「マクロ」タブ > 「継承したマクロとホストマクロ」

テンプレートマクロの設定

[設定] > [テンプレート] > テンプレート名 > 「マクロ」タブ > 「継承したマクロとテンプレートマクロ」

障害発生数上位 100 項目のフィルタ改良

ホスト、ホストグループでのフィルタ、特定の期間によるフィルタ、日、週、月のショートカットリンクが追加されました。

障害発生数上位 100 項目

[レポート] > 障害発生数上位 100 項目

その他

housekeeperの手動実行、自動実行の無効化

Zabbixでは保存期間の過ぎた古いデータを削除するhousekeeper処理が、定期的に自動実行されます。この処理は削除するデータ量によっては負荷が高く、監視や通知に影響を与える場合があります。これを改善するために、housekeeper処理を自動的に行なわず、任意のタイミングで手動実行することができるようになりました。

手動実行するには以下のコマンドを実行します。

```
$ zabbix_server -R housekeeper_execute
zabbix_server [10231]: command sent successfully
```

housekeeperの自動実行を無効化するには、Zabbixサーバの 設定ファイル /etc/zabbix/zabbix_server.conf で「HousekeepingFrequency」を「0」にします。

```
### Option: HousekeepingFrequency
#       How often Zabbix will perform housekeeping procedure (in hours).
#       Housekeeping is removing outdated information from the database.
#       To prevent Housekeeper from being overloaded, no more than 4
times HousekeepingFrequency
#       hours of outdated information are deleted in one housekeeping
cycle, for each item.
#       To lower load on server startup housekeeping is postponed for 30
minutes after server start.
#       With HousekeepingFrequency=0 the housekeeper can be only executed
using the runtime control option.
#       In this case the period of outdated information deleted in one
housekeeping cycle is 4 times the
#       period since the last housekeeping cycle, but not less than 4
hours and not greater than 4 days.
#
# Mandatory: no
# Range: 0-24
# Default:
# HousekeepingFrequency=1
HousekeepingFrequency=0
```

パフォーマンスの改善

- 少量のアイテムから多量の値が取得されたときの処理が改善されました。
- 障害通知を行なうトリガー条件の評価が改善されました。データベースへのアクセス負荷が低減され、効率よく行なわれるようになりました。
- poller がロックする設定キャッシュの数が減りました。高負荷時のパフォーマンスが改善されることが期待されます。
- 内部的なネットワークパフォーマンスが改善されました。

アイテムの追加、改良

`proc.cpu.util` が新しく追加され、Linux や Solaris でプロセスごとの CPU 情報を取得できるようになりました。

その他にもアイテムの追加や改良がされています。

API の追加、改良

`trend.get` でトレンドの取得ができるようになりました。

その他にも関数の追加や改良がされています。