20. Модули

Обзор

Можно улучшить функциональность веб-интерфейса Zabbix, добавив сторонние модули или разработав собственные модули без необходимости изменения исходного кода Zabbix.

Обратите внимание, что код модуля будет работать с теми же привилегиями, что и исходный код Zabbix. Это означает, что:

- Сторонние модули могут быть вредными. Вы должны доверять модулям, которые вы устанавливаете;
- Ошибки в коде стороннего модуля могут привести к сбою интерфейса. Если это произойдет, просто удалите код модуля из внешнего интерфейса. Как только вы перезагрузите интерфейс Zabbix, вы увидите заметку о том, что некоторые модули отсутствуют. Перейдите в Администрирование модулей (в разделе *Администрирование →Общие→ Модули*) и нажмите *Сканировать директорию*, чтобы удалить несуществующие модули из базы данных.

Установка

Пожалуйста, всегда читайте руководство по установке для конкретного модуля. Рекомендуется устанавливать новые модули один за другим, чтобы легко обнаруживать сбои.

Непосредственно перед установкой модуля:

- Убедитесь, что вы загрузили модуль из надежного источника. Установка вредоносного кода может привести к таким последствиям, как потеря данных
- Различные версии одного и того же модуля (один и тот же идентификатор) могут быть установлены параллельно, но одновременно может быть включена только одна версия

Шаги по установке модуля:

- Распакуйте модуль в отдельную папку в папке `modules` веб-интерфейса Zabbix.
- Убедитесь, что папка устанавливаемого модуля содержит как минимум файл manifest.json
- Перейдите к Администрирование модуля и нажмите кнопку *Сканировать директорию*
- Новый модуль появится в списке вместе с описанием, статусом и информацией о версии и авторе модуля.
- Включите модуль, нажав на его статус

Разрешение проблем:

Проблема	**Решение**
Модуль не появился в списке.	Убедитесь, что файл manifest.json присутствует в папке `modules/your-module/` веб-интерфейса Zabbix. Если это так, значит, модуль не соответствует текущей версии Zabbix. Если файл manifest.json не существует, возможно, вы распаковали его не в ту папку.

Проблема	Решение
Веб-интерфейс не запускается.	Код модуля не совместим с текущей версией Zabbix или конфигурацией сервера. Пожалуйста, удалите файлы модуля и перезагрузите интерфейс. Вы увидите уведомление, что некоторые модули отсутствуют. Перейдите в Администрирование модулей и нажмите *Сканировать директорию* еще раз, чтобы удалить несуществующие модули из базы данных.
Появляются сообщения об идентичном пространстве имен, ID или действии.	Новый модуль попытался зарегистрировать пространство имен, ID или действия, которые уже зарегистрированы другим включенным модулем. Отключите конфликтующий модуль (упомянутый в сообщении об ошибке) перед включением нового.
Появляются сообщения о технической ошибке.	Сообщите об ошибках разработчику модуля.

Разработка модулей

Модули написаны на языке PHP. Предпочтительнее использовать шаблон проектирования Model-View-Controller (MVC, «Модель-Вид-Контроллер»), так как он также используется в веб-интерфейсе Zabbix и облегчит разработку. Строгая типизация PHP также приветствуется, но не обязательна.

Обратите внимание, что с модулями вы можете легко добавлять новые пункты меню и соответствующие виды и действия в веб-интерфейс Zabbix. В настоящее время невозможно зарегистрировать новый API или создать новые таблицы базы данных с помощью модулей.

Структура модуля

Каждый модуль представляет собой папку (помещенную в папку `modules`) с вложенными папками, содержащими контроллеры, виды и любой другой код:

```
example_module_directory/           (required)
    manifest.json                   (required)  Metadata and action
definition.
    Module.php                                  Module initialization and
event handling.
    actions/                                    Action controller files.
        SomethingView.php
        SomethingCreate.php
        SomethingDelete.php
        data_export/
            ExportAsXml.php
            ExportAsExcel.php
    views/                                      View files.
        example.something.view.php
        example.something.delete.php
        js/                                     JavaScript files used in
views.
            example.something.view.js.php
    partials/                                   View partial files.
```

```
        example.something.reusable.php
        js/                                     JavaScript files used in
partials.
            example.something.reusable.js.php
```

Как видите, единственным обязательным файлом в директории пользовательских модулей является 'manifest.json'. Модуль не будет регистрироваться без этого файла. «Module.php» отвечает за регистрацию пунктов меню и обработку событий, таких как «onBeforeAction» и «onTerminate». Директории *actions* (действия), *views* (виды) и *partials* (составляющие) содержат код PHP и JavaScript, необходимый для действий модуля.

Соглашение об именах

Прежде чем создавать модуль, важно придти к соглашению об именах для различных элементов модуля, таких как папки и файлы, чтобы придерживаться одной системы организации. Вы также можете найти примеры выше, в разделе Структура модуля.

Элемент	**правила именования**	**Пример**
Папка модуля	Строчные буквы [a-z], нижнее подчеркивание и десятичные цифры	example_v2
Папки действий	Строчные буквы [a-z], нижнее подчеркивание	data_export
Файлы действий	CamelCase, заканчивающийся типом действия	SomethingView.php
Файлы видов и составляющих	Строчные буквы [a-z] Слова, разделенные точкой с префиксом `module.` , за которым следует имя модуля и заканчивающиеся типом действия и расширением .php	module.example.something.view.php
Файлы Javascript	Те же правила, что и для файлов видов и составляющих, за исключением расширения: .js.php .	module.example.something.view.js.php

Обратите внимание, что префикс и включение имени модуля является обязательным для файлов видов и составляющих, за исключением случаев, когда нужно заменить базовые отображаемые элементы или составляющие Zabbix. Однако это правило не распространяется на имена файлов действий.

Подготовка манифеста

Каждый модуль должен иметь файл manifest.json со следующими полями в формате JSON:

Параметр	**Обязательный**	**Тип**	**По умолчанию**	**Описание**
manifest_version	Да	Дробное число	-	Манифестная версия модуля. В настоящее время поддерживается версия **1**.

Параметр	Обязательный	Тип	По умолчанию	Описание
id	Да	Строка	-	ID модуля. Только один модуль с данным ID может быть включен одновременно.
name	Да	Строка	-	Название модуля как отображено в разделе «Администрирование».
version	Да	Строка	-	Версия модуля как отображено в разделе «Администрирование».
namespace	Да	Строка	-	
author	Нет	Строка	""	Автор модуля как отображено в разделе «Администрирование».
url	Нет	Строка	""	URL модуля как отображено в разделе «Администрирование».
description	Нет	Строка	""	Описание модуля как отображено в разделе «Администрирование».
actions	Нет	Объект	{}	Действия для регистрирования с этим модулем. Смотрите «Действия».
config	Нет	Объект	{}	Конфигурация модуля

Для справки см. Пример файла manifest.json в разделе Справочные материалы.

Действия

Модуль будет контролировать действия внешнего интерфейса, определенные в объекте *actions* в файле manifest.json. Таким образом определяются новые действия. Таким же образом вы можете переопределить существующие действия. Каждый ключ действий должен представлять имя действия, а соответствующее значение должно содержать ключи «class» и, по желанию, «layout» и «view».

Одно действие определяется четырьмя составляющими: name, controller, view и layout. Проверка и подготовка данных обычно выполняются в controller, форматирование вывода выполняется в view или в partials, а layout отвечает за оформление страницы такими элементами, как меню, верхний колонтитул, нижний колонтитул и другие. Действия модуля должны быть определены в файле manifest.json как объект *actions*:

Параметр	Обязательный	Тип	По умолчанию	Описание
key	Да	Строка	-	Название действия, строчные буквы [a-z], разделяя слова точкой.
class	Да	Строка	-	Имя класса действия, включая путь к вложенной папке (если используется) в папке 'actions'.
layout	Нет	Строка	"layout.htmlpage"	Макет действия.
view	Нет	Строка	null	Отображение действия.

Существует несколько готовых макетов, таких как «layout.json» или «layout.xml». Они предназначены для действий, которые дают результат, отличный от HTML. Вы можете изучить готовые макеты в директории app/views/ или создать свой собственный.

В некоторых случаях необходимо изменить только виды действия, не изменяя контроллер. В

таком случае просто поместите необходимые элементы отображения и/или составляющие в папку `views` модуля.

Для справки см. пример файла Action controller в разделе Справочные материалы. Также вы можете взять за основу текущие действия исходного кода Zabbix, расположенного в папке app/.

Module.php

Этот необязательный файл PHP отвечает за инициализацию модуля, а также за обработку событий. Предполагается, что в этом файле будет определен класс «Модуль», расширяющий базовый класс «\Core\CModule». Класс Module должен быть определен в пространстве имен, указанном в файле manifest.json.

```
<?php

namespace Modules\Example;
use Core\CModule as BaseModule;

class Module extends BaseModule {
    ...
}
```

Для справки см. пример файла Module.php в разделе Справочные материалы.

Справочные материалы

Этот раздел содержит базовые версии различных элементов модуля, представленных в предыдущих разделах.

manifest.json

```
{
    "manifest_version": 1.0,
    "id": "example_module",
    "name": "Example module",
    "version": "1.0",
    "namespace": "Example",
    "author": "John Smith",
    "url": "http://module.example.com",
    "description": "Short description of the module.",
    "actions": {
        "example.something.view": {
            "class": "SomethingView",
            "view": "module.example.something.view"
        },
        "example.something.create": {
            "class": "SomethingCreate",
```

```
            "layout": null
        },
        "example.something.delete": {
            "class": "SomethingDelete",
            "layout": null
        },
        "example.something.export.xml": {
            "class": "data_export/ExportAsXml",
            "layout": null
        },
        "example.something.export.excel": {
            "class": "data_export/ExportAsExcel",
            "layout": null
        }
    },
    "config": {
        "username": "john_smith"
    }
}
```

Module.php

```
<?php declare(strict_types = 1);

namespace Modules\Example;

use APP;
use CController as CAction;

/**
 * Please see Core\CModule class for additional reference.
 */
class Module extends \Core\CModule {

    /**
     * Initialize module.
     */
    public function init(): void {
        // Initialize main menu (CMenu class instance).
        APP::Component()->get('menu.main')
            ->findOrAdd(_('Reports'))
                ->getSubmenu()
                    ->add((new \CMenuItem(_('Example wide report')))
                        ->setAction('example.report.wide.php')
                    )
                    ->add((new \CMenuItem(_('Example narrow report')))
                        ->setAction('example.report.narrow.php')
                    );
    }
```

```
    /**
     * Event handler, triggered before executing the action.
     *
     * @param CAction $action  Action instance responsible for current
request.
     */
    public function onBeforeAction(CAction $action): void {
    }

    /**
     * Event handler, triggered on application exit.
     *
     * @param CAction $action  Action instance responsible for current
request.
     */
    public function onTerminate(CAction $action): void {
    }
}
```

Action controller

```
<?php declare(strict_types = 1);

namespace Modules\Example\Actions;

use CControllerResponseData;
use CControllerResponseFatal;
use CController as CAction;

/**
 * Example module action.
 */
class SomethingView extends CAction {

    /**
     * Initialize action. Method called by Zabbix core.
     *
     * @return void
     */
    public function init(): void {
        /**
         * Disable SID (Sessoin ID) validation. Session ID validation should
only be used for actions which involde data
         * modification, such as update or delete actions. In such case
Session ID must be presented in the URL, so that
         * the URL would expire as soon as the session expired.
         */
        $this->disableSIDvalidation();
```

```
    }

    /**
     * Check and sanitize user input parameters. Method called by Zabbix
core. Execution stops if false is returned.
     *
     * @return bool true on success, false on error.
     */
    protected function checkInput(): bool {
        $fields = [
            'name'  => 'required|string',
            'email' => 'required|string',
            'phone' => 'string'
        ];

        // Only validated data will further be available using
$this->hasInput() and $this->getInput().
        $ret = $this->validateInput($fields);

        if (!$ret) {
            $this->setResponse(new CControllerResponseFatal());
        }

        return $ret;
    }

    /**
     * Check if the user has permission to execute this action. Method
called by Zabbix core.
     * Execution stops if false is returned.
     *
     * @return bool
     */
    protected function checkPermissions(): bool {
        $permit_user_types = [USER_TYPE_ZABBIX_ADMIN,
USER_TYPE_SUPER_ADMIN];

        return in_array($this->getUserType(), $permit_user_types);
    }

    /**
     * Prepare the response object for the view. Method called by Zabbix
core.
     *
     * @return void
     */
    protected function doAction(): void {
        $contacts = $this->getInput('email');

        if ($this->hasInput('phone')) {
            $contacts .= ', '.$this->getInput('phone');
```

```
        }

        $data = [
            'name' => $this->getInput('name'),
            'contacts' => $contacts
        ];

        $response = new CControllerResponseData($data);

        $this->setResponse($response);
    }
}
```

Action view

```
<?php declare(strict_types = 1);

/**
 * @var CView $this
 */

$this->includeJsFile('example.something.view.js.php');

(new CWidget())
    ->setTitle(_('Something view'))
    ->addItem(new CDiv($data['name']))
    ->addItem(new CPartial('module.example.something.reusable', [
        'contacts' => $data['contacts']
    ]))
    ->show();
```