Интеграция OTRS -> Mattermost -> Telegram

Пример интеграции OTRS и Mattermost основан на примере интеграции OTRS и Slack: https://otrs.ru/forum/viewtopic.php?f=2&t=927&p=25053#p25053

Интеграция протестирована на OTRS версий 5 и 6. Основное отличие последней в необходимости примерения конфигурации из файлов xml. В OTRS 6 конфигурация хранится в базе данных.

Настройка Incoming Webhook в Mattermost

Регистрируемся/логинимся в Mattermost. Переходим: Main Menu --> Integrations --> Incoming Webhooks

Создаем новый Webhook вида: https://mattermosthost/hooks/1234webhookhash67890 , где http - протокол, mattermosthost - хост Mattermost, hooks - адрес API Webhook (выбирается автоматически), 1234webhookhash67890 - хэш нашего Webhook. Все эти параметры выбираются системой автоматически, но зависят от того, каким образом вы залогинены в Mattermost. Единственный выбор предоставленный Вам - это возможность выбрать канал по умолчанию и закрепить Webhook за этим калалом. Я канал выбрал, а закрепление Webhook за этим каналом не делал для того, чтобы иметь возможность отправлять POST запросы на создание сообщений в разные каналы.

Настройка OTRS

Добавление файлов Web Service

В файловую систему OTRS сервера нужно добавить файлы:

- MattermostText.pm (файл с Invoker модулем, который будет отправлять POST-запросы в Mattermost:
- REST.pm (Это модифицированный файл из каталога /opt/otrs/Kernel/GenericInterface/Transport/HTTP)

Модификация последнего необходима для корректной обработки ответа на POST-запросы от Mattermost. Mattermost как и Slack отвечает:

```
"Content-Type" : "text/plain" - в заголовке
и
"ok" - в теле ответа
```

Для модификации создайте директорию и скопируйте файл:

```
mkdir -P /opt/otrs/Custom/Kernel/GenericInterface/Transport/HTTP
cp -p /opt/otrs/Kernel/GenericInterface/Transport/HTTP/REST.pm
/opt/otrs/Custom/Kernel/GenericInterface/Transport/HTTP/REST.pm
```

Далее в скопированном файле с помощью редактора перед строками:

```
    # To convert the data into a hash, use the JSON module.
    my $Result;
```

добавьте строки:

```
    if ($ResponseContent eq 'ok' ) {
        $ResponseContent = '{"text": "ok"}';
    }
```

MattermostText.pm извлеките из архива и положите в каталог файл инвокера 'MattermostText.pm':

```
uzip invoker.zip
mkdir -P /opt/otrs/Custom/Kernel/GenericInterface/Invoker/Mattermost/
cp ./invoker/GenericInterface/Invoker/Mattermost/MattermostText.pm
/opt/otrs/Custom/Kernel/GenericInterface/Invoker/Mattermost/
```

Не забываем о правах. Все файлы и созданные каталоги должны принадлежать пользователю, от имени которого выполняется OTRS:

```
cd /opt/otrs/Custom/Kernel/
sudo chown -R otrs:apache ./GenericInterface
sudo /opt/otrs/bin/otrs.SetPermissions.pl --otrs-user='otrs' --web-group='apache'
```

, где otrs - имя пользователя, от имени которого запущен OTRS, apache - группа пользователя, от имени которого запущен OTRS.

Регистрация нового WEB Service

Для начала необходимо зарегистрировать новый WEB Service. Делается это модификацией файла: /opt/otrs/Kernel/Config/Files/GenericInterface.xml

Для этого сохраним оригинальный файл:

```
cd /opt/otrs/Kernel/Config/Files/
cp -p ./GenericInterface.xml ./GenericInterface.orig
```

В файл /opt/otrs/Kernel/Config/Files/GenericInterface.xml добавим строки:

- Для OTRS версии 5:

```
    <ConfigItem
Name="GenericInterface::Invoker::Module###Mattermost::MattermostText" Required="0"
Valid="0">
        <Description Translatable="1">GenericInterface module registration for the
invoker layer.</Description>
        <Group>GenericInterface</Group>
        <SubGroup>GenericInterface::Invoker::ModuleRegistration</SubGroup>
        <Setting>
            <Hash>
                <Item Key="Name">MattermostText</Item>
                <Item Key="Controller">Mattermost</Item>
                <Item
Key="ConfigDialog">AdminGenericInterfaceInvokerDefault</Item>
            </Hash>
        </Setting>
    </ConfigItem>
```

- Для OTRS версии 6:

```
    <Setting Name="GenericInterface::Invoker::Module###Mattermost::MattermostText"
Required="0" Valid="0">
        <Description Translatable="1">GenericInterface module registration for the
invoker layer.</Description>
        <Navigation>GenericInterface::Invoker::ModuleRegistration</Navigation>
        <Value>
            <Hash>
                <Item Key="Name">MattermostText</Item>
                <Item Key="Controller">Mattermost</Item>
                <Item
Key="ConfigDialog">AdminGenericInterfaceInvokerDefault</Item>
            </Hash>
        </Value>
    </Setting>
```

Я ориентировался на преконфигурированный Web Service с именем TestSimple. Он есть в штатной поставке OTRS. В OTRS 6 для импорта сделанных настроек нужно выполнить:

```
sudo -u otrs /opt/otrs/bin/otrs.Console.pl Maint::Config::Rebuild
```

Делаем "restart httpd":

```
systemctl restart httpd
```

Теперь необходимо включить наш модуль в OTRS, чтобы иметь возможность выбрать его в веб-сервисах. Для этого необходимо зайти в Администрирование/Конф.системы/GenericInterface --> GenericInterfaceInvokerModuleRegistration И поставить галочку напротив сконфигурированного сервиса MattermostText.

Настройка Web Service в OTRS

В административной панели OTRS необходимо создать новый сервис: Admin --> Web Service Management --> Add Web Service С параметрами:

```
General:
    Name: Mattermost
    Debug threshold: Debug
OTRS as requester:
    Settings:
        Network transport: HTTP::REST
    Invokers:
        Name: Mattermost
        CONTROLLER: Mattermost::MattermostText
```

Далее необходимо сконфигурировать Network transport и Invoker. Кнопки Configure появятся сразу после сохранения выбранных параметров.

Network transport

Здесь важно выдержать параметры в точности, как описано. Особо обращаю внимание на то, что даже если вы интегрируетесь через HTTPS, никаких сертификатов загружать не нужно. Важно просто указать протокол в адресе сервера как 'https://' Редиректы инверсных прокси (301 код) REST.pm не умеет обрабатывать или я не умею их готовить.

```
Type: HTTP::REST
Endpoint: https://mattermosthost/hooks
```

```
Controller mapping for Invoker 'MattermostText': 1234webhookhash67890
Valid request command for Invoker 'MattermostText': POST
Default command: POST
Authentication:
Use SSL Options: No
```

Invoker

```
Name: MattermostText
Invoker backend: Mattermost::MattermostText
Mapping for outgoing request data: Simple
Mapping for incoming request data: Simple
Event Triggers:
    EVENT: TicketCreate
    ASYNCHRONOUS: Yes
    CONDITION: No
```

Simple Mapping for Outgoing Data

Если вы планируете отправлять сообщения в несколько разных каналов Mattermost, то здесь нужно сконфигурировать соответствие имен полей и их значений между OTRS и Mattermost. Дело в том, что по умолчанию POST-запрос в теле содержит словарь {ключ: значение}. В нашем случае это будет словарь подобный следующему:

```
{
    "text" :    "#2020033007000174
                Customer: ИмяКастомера
                Queue: queue1
                Priority: 3 normal
                State: new
                Owner: root@localhost
                Responsible: root@localhost
                Changed : 2020-03-30 17:50:44


http://otrs.hostname/otrs/index.pl?Action=AgentTicketZoom;TicketID=73",
    "queue" :   "queue1"
}
```

Для тех кто не знаком с темой полезно будет протестировать обмен запросами с помощью curl или Postman (предварительно замените адрес запроса на полученный в Mattermost Incoming Webhook:

```
curl -i -X POST -H 'Content-Type: application/json' -d '{"text": "[inline
URL](http://yandex.ru)"}' http://mattermosthost:8065/hooks/1234webhookhash67890
```

Последняя пара ключ-значение содержит имя очереди нового тикета. Мы будем её мапить (отображать) в нужный нам канал. В моём случае разные очереди в OTRS обслуживаются разными группами агентов (инженеров). Таким образом, каждая из групп агентов будет читать свой канал в Mattermost (и Telegram). Чтобы настроить отображение очереди OTRS в канал Mattermost нужно настроить Mapping for Outgoing Data в сконфигурированом Invoker'e.

```
Default rule for unmapped keys: Keep (leave unchanged)
Default rule for unmapped values: Keep (leave unchanged)

*Mapping for Key channel*
Key mapping:
Map key: queue
matching the: Exact value(s)
```

```
to new key: channel

Value mapping:
Map value: queue1
matching the: Exact value(s)
to new value: mm_channel1

Map value: queue2
matching the: Exact value(s)
to new value: mm_channel2

Map value: [\w]+
matching the: Regular expression
to new value: mm_default_channel
```

Вы можете настроить свою логику или захардкодить канал. Для этого в файле /opt/otrs/Custom/Kernel/GenericInterface/Invoker/Mattermost/MattermostText.pm нужно поправить строки:

```
# Add field with "queue" name to post-request to Mattermost
# We will convert it to "channel" field in post-request by the "Simple" mapping in
OTRS
#
    $MattermostText{Data}->{queue}=$Ticket{Queue};

#    Instead previus item you can hardcode channel by following
#    $MattermostText{Data}->{channel}="test1";
```

В случае использования $MattermostText->="test1"; вам не нужно настраивать маппинг.

Debugging

В интерфейсе настройки Web Service слева нижняя кнопка "Debugger". При нажатии на нее вы можете посмотреть как обрабатывается POST-запрос после появления нового тикета. Здесь виден и сам сформированный запрос и его отображение в процессе маппинга и ответ на запрос.

Пара слов о Telegram

Для интеграции Mattermost с Telegram используется Mattermost Bridge. Скачать и почитать о его настройке можно на сайте проекта: https://github.com/42wim/matterbridge

Интеграция крайне не затейливая. Для работы с Telegram сильно поможет размещение Mattermost Bridge на сервере за рубежом. Я использую несколько групп в Telegram, каждой из которых соответствует свой канал в Mattermost. Соответственно, разные агенты подписаны на разные группы Telegram для получения сообщений о новых тикетах в разных очередях OTRS.

Успехов.