

Slack Webhook

- 1.
- 2.
- 3.
- 4.
- 5./ API
- 6.
- 7. Just code it!
 - a.
 - b.
 - c.
 - d.
 - e. Felix
 - f.
- 8.

Slack Webhook [Bean Shell](#)

1.

Joget Workflow[Slack](#)

2.

[/ POST](#) [Slack](#)

3.

Slack Webhook[Slack Incoming Webhook](#)

4.

Slack

5./ API

[slack-webhook](#)[Slack](#)

6.

Joget Workflow Source Code

Macbook ProJoget5.0.1

```
- Home
- joget
  - plugins
  - jw-community
    -5.0.1
```

""jw-community"Joget Workflow

"plugins"maven

```
cd joget/plugins/
~/joget/jw-community/5.0.1/wflow-plugin-archetype/create-plugin.sh org.joget slack_webhook 5.0.1
```

shellmaven

```
Define value for property 'version': 1.0-SNAPSHOT: : 5.0.0
[INFO] Using property: package = org.joget
Confirm properties configuration:
groupId: org.joget
artifactId: slack_webhook
version: 5.0.0
package: org.joget
Y: : y
```

"BUILD SUCCESS""plugins""slack_webhook"

IDEmaven [NetBeans](#)

7. Just code it!

a.

"org.joget""SlackWebhookTool"**org.joget.plugin.base.DefaultApplicationPlugin** / POST **org.joget.plugin.base.PluginWebSupport** [Web](#)

b.

AppPluginUtil.getMessage18nMESSAGE_PATH

Implementation of all basic abstract methods

```
package org.joget;

import org.joget.apps.app.service.AppPluginUtil;
import org.joget.apps.app.service.AppUtil;
import org.joget.plugin.base.DefaultApplicationPlugin;
import org.joget.plugin.base.PluginWebSupport;

public class SlackWebhookTool extends DefaultApplicationPlugin implements PluginWebSupport {
    private final static String MESSAGE_PATH = "message/SlackWebhookTool";

    @Override
    public String getName() {
        return "Slack Webhook Tool";
    }

    @Override
    public String getVersion() {
        return "5.0.0";
    }

    @Override
    public String getClassName() {
        return getClass().getName();
    }

    @Override
    public String getLabel() {
        //support i18n
        return AppPluginUtil.getMessage("org.joget.SlackWebhookTool.pluginLabel", getClassName(), MESSAGE_PATH);
    }

    @Override
    public String getDescription() {
        //support i18n
        return AppPluginUtil.getMessage("org.joget.SlackWebhookTool.pluginDesc", getClassName(), MESSAGE_PATH);
    }

    @Override
    public String getPropertyOptions() {
        return AppUtil.readPluginResource(getClass().getName(), "/properties/slackWebhookTool.json", null,
true, MESSAGE_PATH);
    }
}
```

UigetPropertyOptions "/properties/slackWebhookTool.json""slack_webhook / src / main""resources / properties""properties""slackWebhookTool.json"

"@@ message.key @@""i18n"

```
[{
  title : '@@SlackWebhookTool.config@@',
  properties : [{
    name : 'url',
    label : '@@SlackWebhookTool.url@@',
    type : 'textfield',
    required : 'true'
  }],
  {
    label : '@@SlackWebhookTool.from@@',
    type : 'header'
  },
  {
    name : 'username',
    label : '@@SlackWebhookTool.fromUsername@@',
    type : 'textfield',
    value : '@@SlackWebhookTool.fromUsername.value@@'
  },
  {
    name : 'customIcon',
```

```

label : '@@SlackWebhookTool.customIcon@@',
type : 'selectbox',
value : 'joget',
options : [{
  value : '',
  label : '@@SlackWebhookTool.customIcon.none@@'
},
{
  value : 'joget',
  label : '@@SlackWebhookTool.customIcon.joget@@'
},
{
  value : 'url',
  label : '@@SlackWebhookTool.customIcon.url@@'
},
{
  value : 'emoji',
  label : '@@SlackWebhookTool.customIcon.emoji@@'
}]
},
{
  name : 'iconUrl',
  label : '@@SlackWebhookTool.customIcon.url@@',
  type : 'textfield',
  required : 'true',
  control_field: 'customIcon',
  control_value: 'url',
  control_use_regex: 'false'
},
{
  name : 'iconEmoji',
  label : '@@SlackWebhookTool.customIcon.emoji@@',
  type : 'textfield',
  required : 'true',
  control_field: 'customIcon',
  control_value: 'emoji',
  control_use_regex: 'false'
},
{
  label : '@@SlackWebhookTool.to@@',
  type : 'header'
},
{
  name : 'channels',
  label : '@@SlackWebhookTool.channels@@',
  description : '@@SlackWebhookTool.channels.desc@@',
  type : 'textfield'
},
{
  name : 'participantIds',
  label : '@@SlackWebhookTool.participantIds@@',
  description : '@@SlackWebhookTool.participantIds.desc@@',
  type : 'textfield'
},
{
  name : 'usernames',
  label : '@@SlackWebhookTool.usernames@@',
  description : '@@SlackWebhookTool.usernames.desc@@',
  type : 'textfield'
},
{
  name : 'usernameTransform',
  label : '@@SlackWebhookTool.usernameTransform@@',
  description : '@@SlackWebhookTool.usernameTransform.desc@@',
  type : 'textfield',
  value : '@@SlackWebhookTool.usernameTransform.value@@',
  required : 'True'
},
{
  label : '@@SlackWebhookTool.message@@',
  type : 'header'
}

```

```

    },
    {
        name : 'text',
        label : '@@SlackWebhookTool.text@@',
        description : '@@SlackWebhookTool.text.desc@@',
        type : 'codeeditor',
        required : 'True'
    },
    {
        name : 'unfurl_links',
        label : '@@SlackWebhookTool.unfurl_links@@',
        description : '@@SlackWebhookTool.unfurl_links.desc@@',
        type : 'checkbox',
        value : 'true',
        options : [{
            value : 'true',
            label : ''
        }]
    },
    {
        name : 'unfurl_media',
        label : '@@SlackWebhookTool.unfurl_media@@',
        description : '@@SlackWebhookTool.unfurl_media.desc@@',
        type : 'checkbox',
        value : 'true',
        options : [{
            value : 'true',
            label : ''
        }]
    },
    ],
    buttons : [{
        name : 'sendTestMessage',
        label : '@@SlackWebhookTool.sendTestMessage@@',
        ajax_url : '[CONTEXT_PATH]/web/json/app[APP_PATH]/plugin/org.joget.SlackWebhookTool/service?
action=sendTestMessage',
        fields : ['url'],
        addition_fields : [
            {
                name : 'testChannel',
                label : '@@SlackWebhookTool.sendTestMessage.testChannel@@',
                type : 'textfield'
            }
        ]
    }
    ],
    },
    {
        title : '@@SlackWebhookTool.attachment@@',
        properties : [{
            name:'fallback',
            label:'@@SlackWebhookTool.attachment.fallback@@',
            description:'@@SlackWebhookTool.attachment.fallback.desc@@',
            type:'textfield',
            value: '@@SlackWebhookTool.attachment.fallback.value@@',
            required : 'True'
        },
        {
            name:'color',
            label:'@@SlackWebhookTool.attachment.color@@',
            description:'@@SlackWebhookTool.attachment.color.desc@@',
            type:'textfield'
        },
        {
            name:'pretext',
            label:'@@SlackWebhookTool.attachment.pretext@@',
            description:'@@SlackWebhookTool.attachment.pretext.desc@@',
            type:'textfield'
        },
        {
            name:'author_name',
            label:'@@SlackWebhookTool.attachment.author_name@@',
            description:'@@SlackWebhookTool.attachment.author_name.desc@@',

```

```

        type: 'textfield'
    },
    {
        name: 'author_link',
        label: '@@SlackWebhookTool.attachment.author_link@@',
        description: '@@SlackWebhookTool.attachment.author_link.desc@@',
        type: 'textfield'
    },
    {
        name: 'author_icon',
        label: '@@SlackWebhookTool.attachment.author_icon@@',
        description: '@@SlackWebhookTool.attachment.author_icon.desc@@',
        type: 'textfield'
    },
    {
        name: 'attachment_title',
        label: '@@SlackWebhookTool.attachment.title@@',
        description: '@@SlackWebhookTool.attachment.title.desc@@',
        type: 'textfield'
    },
    {
        name: 'attachment_title_link',
        label: '@@SlackWebhookTool.attachment.title_link@@',
        description: '@@SlackWebhookTool.attachment.title_link.desc@@',
        type: 'textfield'
    },
    {
        name: 'attachment_text',
        label: '@@SlackWebhookTool.attachment.text@@',
        description: '@@SlackWebhookTool.attachment.text.desc@@',
        type: 'textfield'
    },
    {
        name: 'fields',
        label: '@@SlackWebhookTool.attachment.fields@@',
        description: '@@SlackWebhookTool.attachment.fields.desc@@',
        type: 'grid',
        columns: [{
            key: 'title',
            label: '@@SlackWebhookTool.attachment.fields.title@@'
        },
        {
            key: 'value',
            label: '@@SlackWebhookTool.attachment.fields.value@@'
        },
        {
            key: 'short',
            label: '@@SlackWebhookTool.attachment.fields.short@@',
            options: [
                {
                    value: 'false',
                    label: '@@SlackWebhookTool.attachment.fields.short.no@@'
                },
                {
                    value: 'true',
                    label: '@@SlackWebhookTool.attachment.fields.short.yes@@'
                }
            ]
        }
    ]
    },
    {
        name: 'image_url',
        label: '@@SlackWebhookTool.attachment.image_url@@',
        description: '@@SlackWebhookTool.attachment.image_url.desc@@',
        type: 'textfield'
    },
    {
        name: 'thumb_url',
        label: '@@SlackWebhookTool.attachment.thumb_url@@',
        description: '@@SlackWebhookTool.attachment.thumb_url.desc@@',
        type: 'textfield'
    }
}
}]
}]

```

```

private SlackApi api = null;

@Override
public Object execute(Map props) {
    final Set<String> channelList = new HashSet<String>();

    String channels = getPropertyString("channels");
    if (!channels.isEmpty()) {
        String[] temp = channels.split(";");
        for (String t : temp) {
            if (t.startsWith("#")) {
                channelList.add(t);
            }
        }
    }

    WorkflowAssignment wfAssignment = (WorkflowAssignment) getProperty("workflowAssignment");

    String participantIds = getPropertyString("participantIds");
    if (wfAssignment != null && !participantIds.isEmpty()) {
        String[] temp = participantIds.split(";");
        if (temp.length > 0) {
            WorkflowManager workflowManager = (WorkflowManager) AppUtil.getApplicationContext().getBean(
("workflowManager");
            WorkflowProcess process = workflowManager.getProcess(wfAssignment.getProcessDefId());
            for (String pid : temp) {
                pid = pid.trim();
                if (!pid.isEmpty()) {
                    Collection<String> userList = WorkflowUtil.getAssignmentUsers(process.getPackageId(),
wfAssignment.getProcessDefId(), wfAssignment.getProcessId(), wfAssignment.getProcessVersion(), wfAssignment.
getActivityId(), "", pid);
                    if (userList != null && userList.size() > 0) {
                        for (String username : userList) {
                            channelList.add(getSlackUsername(username, wfAssignment));
                        }
                    }
                }
            }
        }
    }

    String usernames = getPropertyString("usernames");
    if (!usernames.isEmpty()) {
        String[] temp = usernames.split(";");
        for (String username : temp) {
            channelList.add(getSlackUsername(username, wfAssignment));
        }
    }

    Thread sendMessageThread = new PluginThread(new Runnable() {
        public void run() {
            try {
                LogUtil.info(SlackWebhookTool.class.getName(), "SlackWebhookTool: Sending message to=" +
channelList.toString());
                sendMessage(channelList);
                LogUtil.info(SlackWebhookTool.class.getName(), "SlackWebhookTool: Sending message completed
to=" + channelList.toString());
            } catch (Exception ex) {
                LogUtil.error(SlackWebhookTool.class.getName(), ex, "");
            }
        }
    });

    sendMessageThread.setDaemon(true);
    sendMessageThread.start();

    return null;
}

```

```

    }

    protected SlackApi getApi() {
        if (api == null) {
            api = new SlackApi(getPropertyString("url"));
        }
        return api;
    }

    protected String getSlackUsername(String username, WorkflowAssignment assignment) {
        String syntax = getPropertyString("usernameTransform");
        syntax = syntax.replaceAll(StringUtil.escapeRegex("{username}"), StringUtil.escapeRegex(username));
        return AppUtil.processHashVariable(syntax, assignment, null, null);
    }

    protected void sendMessage(Set<String> channels) {
        SlackMessage message = createMessage();
        if (channels != null && !channels.isEmpty()) {
            for (String channel : channels) {
                if (!channel.isEmpty()) {
                    sendMessage(channel, message);
                }
            }
        } else {
            sendMessage(null, message);
        }
    }

    protected void sendMessage(String channel, SlackMessage message) {
        if (channel != null && !channel.isEmpty()) {
            message.setChannel(channel);
        }

        getApi().call(message);
    }

    protected SlackMessage createMessage() {
        SlackMessage message = new SlackMessage(getPropertyString("text"));

        String username = getPropertyString("username");
        if (!username.isEmpty()) {
            message.setUsername(username);
        }

        String customIcon = getPropertyString("customIcon");
        if (!customIcon.isEmpty()) {
            if ("joget".equals(customIcon)) {
                HttpServletRequest request = WorkflowUtil.getHttpServletRequest();
                if (request != null) {
                    String url = request.getScheme() + "://" + request.getServerName() + ":" + request.
getServerPort() + request.getContextPath() + "/images/v3/logo.png";
                    message.setIcon(url);
                }
            } else if ("url".equals(customIcon)) {
                message.setIcon(getPropertyString("iconUrl"));
            } else {
                message.setIcon(getPropertyString("iconEmoji"));
            }
        }

        message.setUnfurlLinks("true".equalsIgnoreCase(getPropertyString("unfurl_links")));
        message.setUnfurlMedia("true".equalsIgnoreCase(getPropertyString("unfurl_media")));

        boolean hasAttachment = false;
        SlackAttachment atachment = new SlackAttachment();

        String fallback = getPropertyString("fallback");
        if (!fallback.isEmpty()) {
            atachment.setFallback(fallback);
        }
    }

```



```

String color = getPropertyString("color");
if (!color.isEmpty()) {
    atachment.setColor(color);
}

String pretext = getPropertyString("pretext");
if (!pretext.isEmpty()) {
    atachment.setPretext(pretext);
    hasAttachment = true;
}

String author_name = getPropertyString("author_name");
if (!author_name.isEmpty()) {
    atachment.setAuthorName(author_name);
}

String author_link = getPropertyString("author_link");
if (!author_link.isEmpty()) {
    atachment.setAuthorLink(author_link);
}

String author_icon = getPropertyString("author_icon");
if (!author_icon.isEmpty()) {
    atachment.setAuthorIcon(author_icon);
}

String attachment_title = getPropertyString("attachment_title");
if (!attachment_title.isEmpty()) {
    atachment.setTitle(attachment_title);
    hasAttachment = true;
}

String attachment_title_link = getPropertyString("attachment_title_link");
if (!attachment_title_link.isEmpty()) {
    atachment.setTitleLink(attachment_title_link);
}

String attachment_text = getPropertyString("attachment_text");
if (!attachment_text.isEmpty()) {
    atachment.setText(attachment_text);
    hasAttachment = true;
}

String image_url = getPropertyString("image_url");
if (!image_url.isEmpty()) {
    atachment.setImageUrl(image_url);
    hasAttachment = true;
}

String thumb_url = getPropertyString("thumb_url");
if (!thumb_url.isEmpty()) {
    atachment.setThumbUrl(thumb_url);
    hasAttachment = true;
}

Object[] fields = null;
Object fieldsProperty = getProperty("fields");
if (fieldsProperty != null && fieldsProperty instanceof Object[]){
    fields = (Object[]) fieldsProperty;
}

if (fields != null && fields.length > 0) {
    for (Object o : fields) {
        Map mapping = (HashMap) o;
        String title = mapping.get("title").toString();
        String value = mapping.get("value").toString();
        String isShort = mapping.get("short").toString();
        SlackField field = new SlackField();
        field.setTitle(title);
        field.setValue(value);
        field.setShorten("true".equalsIgnoreCase(isShort));
    }
}

```

```

        atachment.addFields(field);
        hasAttachment = true;
    }
}

if (hasAttachment) {
    message.addAttachments(atachment);
}

return message;
}

```

webServiceAPI

```

    public void webService(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        boolean isAdmin = WorkflowUtil.isCurrentUserInRole(WorkflowUserManager.ROLE_ADMIN);
        if (!isAdmin) {
            response.sendError(HttpServletResponse.SC_UNAUTHORIZED);
            return;
        }

        String action = request.getParameter("action");
        if ("sendTestMessage".equals(action)) {
            String message = "";
            try {
                AppDefinition appDef = AppUtil.getCurrentAppDefinition();

                String url = AppUtil.processHashVariable(request.getParameter("url"), null, null, null, appDef);
                String testChannel = AppUtil.processHashVariable(request.getParameter("testChannel"), null,
null, null, appDef);

                setProperty("url", url);
                setProperty("text", AppPluginUtil.getMessage("SlackWebhookTool.testMessage", getClassName(),
MESSAGE_PATH));

                if (testChannel != null && !testChannel.isEmpty()) {
                    Set<String> channels = new HashSet<String>();
                    channels.add(testChannel);
                    sendMessage(channels);
                } else {
                    sendMessage(null);
                }

                message = AppPluginUtil.getMessage("SlackWebhookTool.sendTestMessage.success", getClassName(),
MESSAGE_PATH);
            } catch (Exception e) {
                LogUtil.error(this.getClassName(), e, "Fail to send Test Message to Slack");
                message = AppPluginUtil.getMessage("SlackWebhookTool.sendTestMessage.fail", getClassName(),
MESSAGE_PATH) + "\n" + StringEscapeUtils.escapeJavaScript(e.getMessage());
            }
            try {
                JSONObject jsonObject = new JSONObject();
                jsonObject.accumulate("message", message);
                jsonObject.write(response.getWriter());
            } catch (Exception e) {
                //ignore
            }
        } else {
            response.setStatus(HttpServletResponse.SC_NO_CONTENT);
        }
    }
}

```

c.

POM"jsp-api""slack-webhook"

```

<!-- Change plugin specific dependencies here -->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jsp-api</artifactId>
  <version>2.0</version>
</dependency>
<dependency>
  <groupId>net.gpedro.integrations.slack</groupId>
  <artifactId>slack-webhook</artifactId>
  <version>1.1.1</version>
</dependency>
<!-- End change plugin specific dependencies here -->

```

d.

```
getLabel(getDescription()18n18n
```

```
"slack_webhook / src / main""resources / message""SlackWebhookTool.properties"
```

```

org.joget.SlackWebhookTool.pluginLabel=Slack Webhook Tool
org.joget.SlackWebhookTool.pluginDesc=Used to send message to Slack through incoming webhook
SlackWebhookTool.config=Configure Slack Webhook Tool
SlackWebhookTool.url=Webhook URL
SlackWebhookTool.from=From
SlackWebhookTool.fromUsername=Username
SlackWebhookTool.fromUsername.value=Joget Workflow
SlackWebhookTool.customIcon=Custom Icon
SlackWebhookTool.customIcon.none=None
SlackWebhookTool.customIcon.joget=Joget Workflow Logo
SlackWebhookTool.customIcon.url=Image URL
SlackWebhookTool.customIcon.emoji=Emoji Code
SlackWebhookTool.to=To
SlackWebhookTool.channels=Channels
SlackWebhookTool.channels.desc=Separated by semicolon(;). E.g. #general;#info
SlackWebhookTool.participantIds=Participant Ids
SlackWebhookTool.participantIds.desc=Separated by semicolon(;). E.g. applicant;approver
SlackWebhookTool.usernames=Usernames
SlackWebhookTool.usernames.desc=Separated by semicolon(;). E.g. admin;cat
SlackWebhookTool.usernameTransform=Transform username to Slack username
SlackWebhookTool.usernameTransform.desc=Hash Variable can be used to transform username to Slack username. Eg.
@#form.slack.username[{username}]#
SlackWebhookTool.usernameTransform.value=@{username}
SlackWebhookTool.message=Message
SlackWebhookTool.text=Text
SlackWebhookTool.text.desc=Refer to <a href="https://api.slack.com/docs/formatting" target="_blank">Slack
Message Formatting</a>.
SlackWebhookTool.unfurl_links=Unfurling Links
SlackWebhookTool.unfurl_links.desc=Automatically find URLs in a message and create attachments based on the
content of those URLs
SlackWebhookTool.unfurl_media=Unfurling Media
SlackWebhookTool.unfurl_media.desc=Automatically find Media URLs in a message and create attachments based on
the media of those URLs
SlackWebhookTool.attachment=Attachment
SlackWebhookTool.attachment.fallback=Fallback
SlackWebhookTool.attachment.fallback.desc=A plain-text summary of the attachment. This text will be used in
clients that don't show formatted text (eg. IRC, mobile notifications) and should not contain any markup.
SlackWebhookTool.attachment.fallback.value=Attachment fail to display.
SlackWebhookTool.attachment.color=Color
SlackWebhookTool.attachment.color.desc=An optional value that can either be one of good, warning, danger, or
any hex color code (eg. #439FE0). This value is used to color the border along the left side of the message
attachment.
SlackWebhookTool.attachment.pretext=Pretext
SlackWebhookTool.attachment.pretext.desc=This is optional text that appears above the message attachment block.
SlackWebhookTool.attachment.author_name=Author Name
SlackWebhookTool.attachment.author_name.desc=Small text used to display the author's name.
SlackWebhookTool.attachment.author_link=Author Link
SlackWebhookTool.attachment.author_link.desc=A valid URL that will hyperlink the Author Name text mentioned
above. Will only work if Author Name is present.

```

SlackWebhookTool.attachment.author_icon=Author Icon
 SlackWebhookTool.attachment.author_icon.desc=A valid URL that displays a small 16x16px image to the left of the Author Name text. Will only work if Author Name is present.
 SlackWebhookTool.attachment.title=Attachment Title
 SlackWebhookTool.attachment.title.desc=The Attachment Title is displayed as larger, bold text near the top of a message attachment.
 SlackWebhookTool.attachment.title_link=Attachment Title Link
 SlackWebhookTool.attachment.title_link.desc=By passing a valid URL in the Attachment Title Link parameter (optional), the Attachment Title text will be hyperlinked.
 SlackWebhookTool.attachment.text=Text
 SlackWebhookTool.attachment.text.desc=This is the main text in a message attachment, and can contain standard message markup. The content will automatically collapse if it contains 700+ characters or 5+ linebreaks, and will display a "Show more..." link to expand the content.
 SlackWebhookTool.attachment.fields=Fields
 SlackWebhookTool.attachment.fields.desc=Fields will be displayed in a table inside the message attachment.
 SlackWebhookTool.attachment.fields.title=Title
 SlackWebhookTool.attachment.fields.value=Value
 SlackWebhookTool.attachment.fields.short=Short
 SlackWebhookTool.attachment.fields.short.no=No
 SlackWebhookTool.attachment.fields.short.yes=Yes
 SlackWebhookTool.attachment.image_url=Image URL
 SlackWebhookTool.attachment.image_url.desc=A valid URL to an image file that will be displayed inside a message attachment. We currently support the following formats: GIF, JPEG, PNG, and BMP. Large images will be resized to a maximum width of 400px or a maximum height of 500px, while still maintaining the original aspect ratio.
 SlackWebhookTool.attachment.thumb_url=Thumbnail URL
 SlackWebhookTool.attachment.thumb_url.desc=A valid URL to an image file that will be displayed as a thumbnail on the right side of a message attachment. We currently support the following formats: GIF, JPEG, PNG, and BMP. The thumbnail's longest dimension will be scaled down to 75px while maintaining the aspect ratio of the image. The filesize of the image must also be less than 500 KB.
 SlackWebhookTool.sendTestMessage=Send Test Message
 SlackWebhookTool.sendTestMessage.testChannel=Test Channel
 SlackWebhookTool.sendTestMessage.success=Test message sent.
 SlackWebhookTool.sendTestMessage.fail=Fail to sent test message. Error:
 SlackWebhookTool.testMessage=Test Message

e. Felix

ActivatorFelix

```
public void start(BundleContext context) {
    registrationList = new ArrayList<ServiceRegistration>();
    //Register plugin here
    registrationList.add(context.registerService(SlackWebhookTool.class.getName(), new SlackWebhookTool(),
null));
}
```

f.

"slack_webhook / target""slack_webhook-5.0.0.jar"

jar jar

Filter by Type		Process Tool	
<input type="checkbox"/> Plugin Name	Plugin Description	Plugin Version	
<input type="checkbox"/> SOAP Tool	Reads a WSDL URL, and	5.0.0	
<input type="checkbox"/> Slack Webhook Tool	Used to send message to	5.0.0	

Slack Webhook Tool

Map Tools to Plugins - Tool 1 (tool1)



Plugin Name	Plugin Description	Plugin Version	
SOAP Tool	Reads a WSDL URL, and i	5.0.0	
Slack Webhook Tool	Used to send message to s	5.0.0	

10 Page 2 of 2 Displaying 11 to 12 of 12 items

SlackIncoming Webhooks

1. your_team.slack.com/services/new
2. WebHook Add
3. Add Incoming WebHooks Integration
4. WebHook " Webhook URL "

Slack Webhook

Configure Slack Webhook Tool

Configure Slack Webhook Tool > Attachment

Webhook URL * <https://hooks.slack.com/services/T02TDBDDW/B0S43NK5K/L>

From

Username

Custom Icon

To

Channels

Participant Ids

Username

Transform username to Slack username *

Message

Text *

1	#form.slack.text#
---	-------------------

< Prev Next >

Send Test Message Submit

Attachment

Configure Slack Webhook Tool > Attachment

Fallback

A plain-text summary of the attachment. This text will be used in clients that don't show formatted text (eg. IRC, mobile notifications) and should not contain any markup.

Attachment fail to display.

Color

good

Pretext

Author Name

Author Link

Author Icon

Attachment Title

The Attachment Title is displayed as larger, bold text near the top of a message attachment.

Received Data

Attachment Title Link

Text

Fields


Title	Value	Short
name	#form.slack.name#	Yes
email	#form.slack.email#	Yes

< Prev

Next >

Submit

Slack



Joget Workflow

BOT

3:44 PM

This is a test message

Received Data

name

Joget

email

info@joget.org

8.

slack_webhook_src.zip

jarhttp://marketplace.joget.org/