



Quality **Forward**

BTS 連携マニュアル

VERISERVE

最終更新日 : 2020/10/28

目次

第 1 章	BTS 連携を行う	2
1.1.	Redmine と連携する	2
1.1.1.	Redmine のベース URL を設定する	3
1.1.2.	新規チケット作成画面の URL	4
1.1.3.	バグ一覧取得用の URL を設定する	4
1.1.4.	最近のインシデント取得用 URL を設定する	7
1.2.	JIRA サーバー型と連携する	7
1.2.1.	ユーザ名とパスワードを入力する	8
1.2.2.	URL・コンテキストパスを設定する	8
1.2.3.	新規チケット作成画面の URL を設定する	9
1.2.4.	JQL を設定する	11
1.2.5.	バグやクローズの文字列を設定する	12
1.3.	JIRA クラウド型と連携する	14
1.3.1.	ユーザ名とパスワードを入力する	14
1.3.2.	新規チケット作成画面の URL を設定する	15
1.3.3.	JQL を設定する	19
1.3.4.	バグやクローズの文字列を設定する	20
第 2 章	よくある質問と回答	21
2.1.	Q: BTS 疎通ができているか確認したい	21
2.2.	Q: BTS 疎通ができない	22
2.3.	Q: BTS の件数がグラフに反映されない	22
2.4.	Q: BTS の Open/Close の件数がレポートに反映されるのはいつですか？	23
2.5.	Q: 過去の BTS の Open/Close 数を修正したい	23
2.6.	Q: JIRA のクラウド型を使用したい	24
2.7.	Q: Redmine で BTS 疎通ができない	24
2.8.	Q: BTS で集計されるチケット範囲を絞りたい	25
2.9.	Q: チケットを CLOSE にしたのに欠陥 OPEN 数が変わらない	27

第1章 BTS 連携を行う

ご使用の BTS を選択し、レポート機能と連携設定することができます。

社内ネットワークを使用されている場合、外部のシステムである QualityForward からのアクセスが弾かれてしまう場合がございます。その場合は以下の IP アドレスからのアクセスを許可していただく必要がございます。

QualityForward IP アドレス

13.112.115.12

13.113.53.12

52.197.246.217

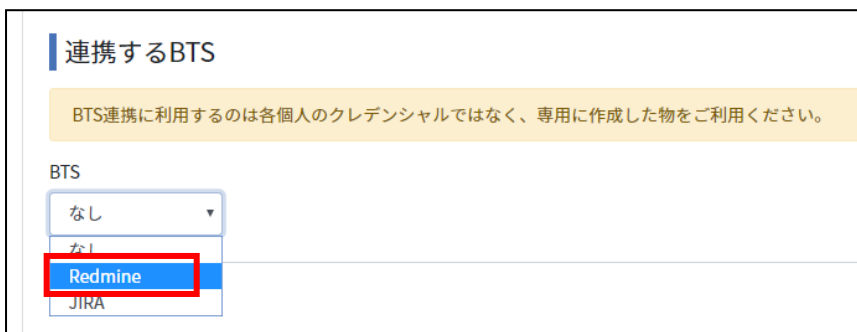
52.197.44.200

18.177.168.126

52.69.201.102

1.1. Redmine と連携する

連携する BTS で「Redmine」を選択すると Redmine に連携するための設定項目が表示されます。



連携するBTS

BTS連携に利用するのは各個人のクレデンシャルではなく、専用に作成した物をご利用ください。

BTS

- なし
- なし
- Redmine**
- JIRA

連携するBTS

BTS連携に利用するのは各個人のクレデンシャルではなく、専用に作成した物をご利用ください。

BTS

Redmine

ベースURL

例：https://xxx/?key=yyyy

●レポート画面での redmine へのリンクなどに利用します

新規チケット作成画面のURL

例：https://xxx/projects/test/issues/new

●テストサイクル実行画面からBTSへの起票を行う際に利用します。

バグ曲線、グラフデータ取得用URL

例：https://xxx/projects/test/issues.json?key=yyyy&query_id=x

●バグ情報の取得に利用します。事前にすべてのチケットのjsonが取得できることをご確認ください

最近のインシデント取得用URL

例：https://xxx/projects/test/issues.json?key=yyyy&query_id=y

●レポート画面での最近のインシデント情報の表示に利用します。事前にレポート画面に表示したい内容のjsonが取得できることをご確認ください

登録する

1.1.1. Redmine のベース URL を設定する

Redmine のベース URL の入力を行います。ベース URL はバグの優先度設定の取得、および「最近のインシデント一覧」のチケットのリンク生成のため利用します。

ベース URL は、Redmine のホームの URL です。

Redmine のルートにあたる URL を指し、 [https://xxxxxx.xxxx/] の場合と [https://xxxxxx.xxxx/redmine/] である場合の 2 パターンあります。URL の後には「?key=API キー」を指定します。

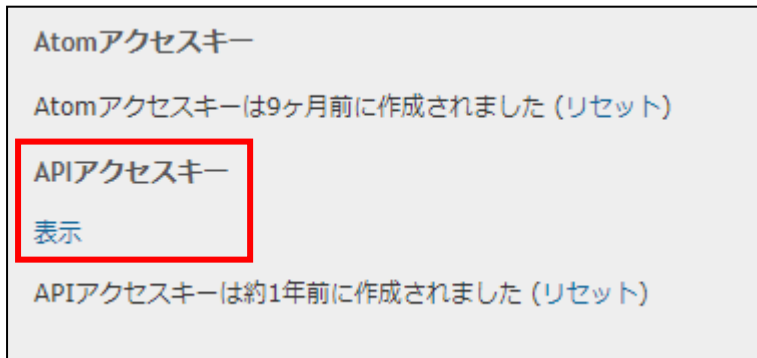
ベースURL

例：https://xxx/?key=yyyy

●レポート画面での redmine へのリンクなどに利用します

API キーは Redmine の個人設定から取得することができます。手順は以下の通りです。

- (1) Redmine にログインし、画面右上の個人設定を開きます。
- (2) API アクセスキーの表示をクリックすると API キーが表示されます。



1.1.2. 新規チケット作成画面の URL

新規チケット作成画面の URL を設定すると、テストサイクルでテストを実行中に右クリックから簡単にチケットへの起票を行うことができます。

チケットの起票画面の URL を記載してください。

例) <https://xxx/projects/test/issue/new>

1.1.3. バグ一覧取得用の URL を設定する

バグ取得用の URL の入力を行います。バグ曲線を描画するため、指定された範囲におけるバグの OPEN/CLOSE 数の取得に利用します。

バグ曲線、グラフデータ取得用URL

例:

●バグ情報の取得に利用します。事前にすべてのチケットのjsonが取得できることをご確認ください

- (1) Redmine の個人設定から API キーを取得します。

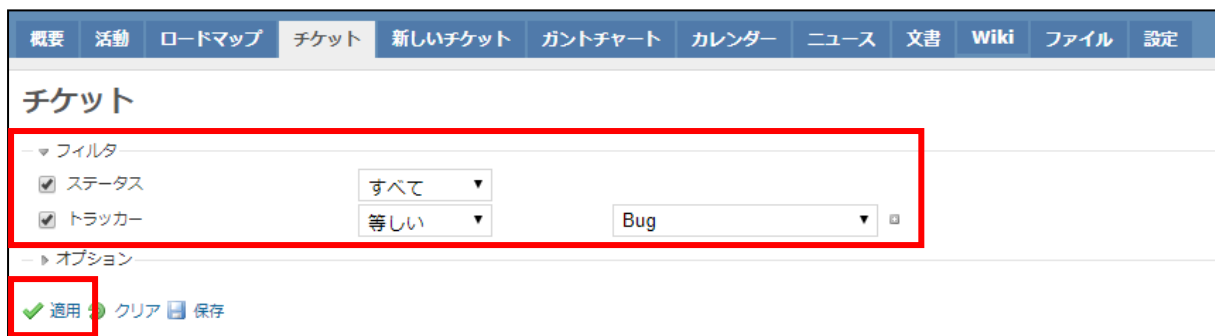
(2) Redmine の対象のプロジェクト開き、URL を取得します。

例) <https://xxx.xxxx/projects/xxxx/>

(3) 手順(2)で取得した URL に「issues.json?key=API キー」を追加します。

例) <https://xxx.xxxx/projects/xxxx/issues.json?key=xxxx>

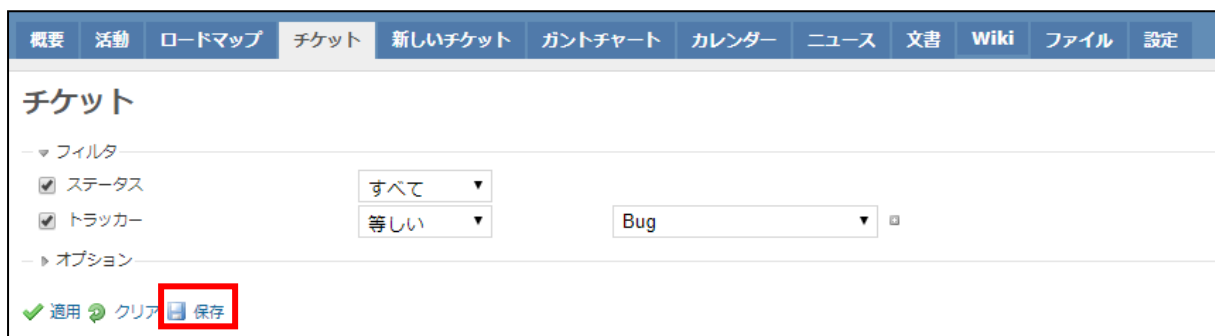
(4) Redmine で対象チケットの絞り込みを行います。バグ一覧を取得するために「すべて」の「バグ」を対象にし、適用ボタンを押します。



The screenshot shows the Redmine 'チケット' (Tickets) page. The navigation bar includes '概要', '活動', 'ロードマップ', 'チケット', '新しいチケット', 'ガントチャート', 'カレンダー', 'ニュース', '文書', 'Wiki', 'ファイル', and '設定'. The main content area is titled 'チケット' and contains a filter section. The filter section has a dropdown arrow on the left. Below it, there are two checked checkboxes: 'ステータス' (Status) and 'トラッカー' (Tracker). The 'ステータス' dropdown is set to 'すべて' (All) and the 'トラッカー' dropdown is set to 'Bug'. There is also a text input field containing 'Bug'. Below the filter section, there is an 'オプション' (Options) section with three buttons: '適用' (Apply), 'クリア' (Clear), and '保存' (Save). The '適用' button is highlighted with a red box.

※ここでは「すべて」の「バグ」を対象にしていますが、フィルタ条件はプロジェクト方針に合わせて自由に設定していただけます。

(5) すべてのバグの一覧が表示されたら保存ボタンを押します。



This screenshot is identical to the previous one, showing the Redmine 'チケット' page with the filter section and the '適用' (Apply) button highlighted. In this screenshot, the '保存' (Save) button is highlighted with a red box.

(6) 新しいクエリに名前を付けて保存します。

新しいクエリ

名称

表示 自分のみ
 すべてのユーザー
 次のロールのみ:
 Manager
 Developer
 Reporter

全プロジェクト向け

オプション

デフォルトの項目

グループ条件

表示 説明
合計 予定工数 作業時間

フィルタ

ステータス
 トラッカー

ソート条件

1:
2:
3:

- (7) チケット一覧右側のカスタムクエリ一覧に手順(6)で作成したクエリが表示されます。作成したクエリ名をクリックします。

チケット

[すべてのチケットを表示](#)

[サマリー](#)

[カレンダー](#)

[ガントチャート](#)

[インポート](#)

カスタムクエリ

- (8) URL の最後にクエリ ID が表示されるので、「query_id=xx」をコピーします。

`/issues?query_id=25`

- (9) 手順(3)までで作成した URL の最後に手順(8)の「&query_id=xx」を入力します。
例)https://xxx.xxxx/projects/xxxx/issues.json?key=xxxx&query_id=xx

- (10) 手順(9)でできた URL をブラウザのアドレス欄に直接入力します。json の取得が確認できたら URL を登録欄に入力し、登録ボタンを押します。

1.1.4. 最近のインシデント取得用 URL を設定する

最近のインシデント取得用 URL 「最近のインシデント一覧」「優先度別グラフ」「ステータス別グラフ」画面を生成するため、指定された範囲のバグのタイトル、優先度、ステータスを取得します。

最近のインシデント取得用URL

例：https://xxx/projects/test/issues.json?key=yyyy&query_id=y

●レポート画面での最近のインシデント情報の表示に利用します。事前にレポート画面に表示したい内容のjsonが取得できることをご確認ください

※URL の取得は手順 [1.1.3](#) と同様に行います。

※バグ一覧取得用 URL と最近のインシデント取得用 URL が個別に設定可能なのは、プロジェクトによって取得したい先のフィルタが異なる場合にも対応するためです。
集計したいバグの範囲が同一の場合には、同一の指定で問題ありません。

1.2. JIRA サーバー型と連携する

連携する BTS で「JIRA」を選択すると JIRA に連携するための設定項目が表示されます。

連携するBTS

BTS連携に利用するのは各個人のクレデンシャルではなく、専用に作成した物をご利用ください。

BTS

なし ▼

なし

Admin

JIRA

1.2.1. ユーザ名とパスワードを入力する

情報を取得するために、JIRA に登録済みのユーザ名とパスワードを入力します。

ユーザ名
<input type="text" value="ユーザ名"/>
パスワード
<input type="password" value="パスワード"/>

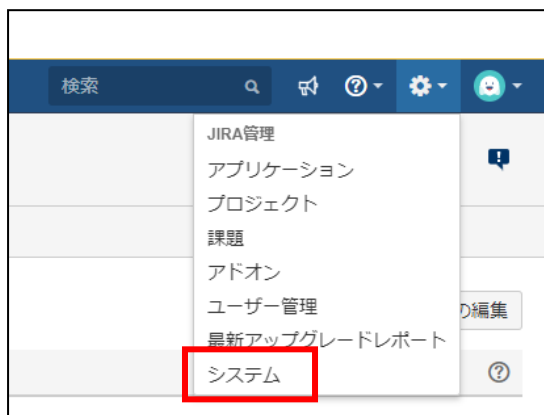
1.2.2. URL・コンテキストパスを設定する

取得したい課題の登録されたプロジェクトを含む JIRA の URL を設定します。コンテキストパスは JIRA 側で設定を行っている場合にのみ入力してください。

※コンテキストパスは「/xxx」の形式で入力してください。

JIRAのURL
<input type="text" value="https://xxx.jp/"/>
コンテキストパス
<input type="text" value="例：/jira"/>
<small>● JIRA側で設定している場合にのみ入力してください</small>

(1) JIRA の管理メニューからシステムを選択します。



(2) 一般設定を開きます。

The screenshot shows the JIRA Administration interface. At the top, there are navigation tabs: 'ダッシュボード', 'プロジェクト', '課題', 'ボード', and '作成'. Below this is a search bar for '管理' (Administration) with the text 'JIRA 管理の検索'. A horizontal menu contains 'アプリケーション', 'プロジェクト', '課題', 'アドオン', 'ユーザー管理', '最新アップグレードレポート', and 'システム'. On the left, a sidebar lists various administration tools, with '一般設定' (General Settings) highlighted by a red box. The main content area is titled '設定' (Settings) and contains a table of configuration options.

設定	
一般設定	
タイトル	QFJIRA
モード	非公開
認証の最大試行回数	3
サインアップ時に CAPTCHA を使用	オフ
ベース URL	http://jira-eval.vtsuite.net:8080
メールの差出人	\${fullname} (JIRA)
概要	
多言語対応	

(3) 一般設定内にあるベース URL をコピーし、JIRA の URL に入力します。

This is a close-up view of the '設定' (Settings) page, specifically the '一般設定' (General Settings) section. The 'ベース URL' (Base URL) field is highlighted with a red box, showing the value 'http://jira-eval.vtsuite.net:8080'. Other settings visible include 'タイトル' (Title) as 'QFJIRA', 'モード' (Mode) as '非公開' (Private), and 'サインアップ時に CAPTCHA を使用' (Use CAPTCHA at sign-up) as 'オフ' (Off).

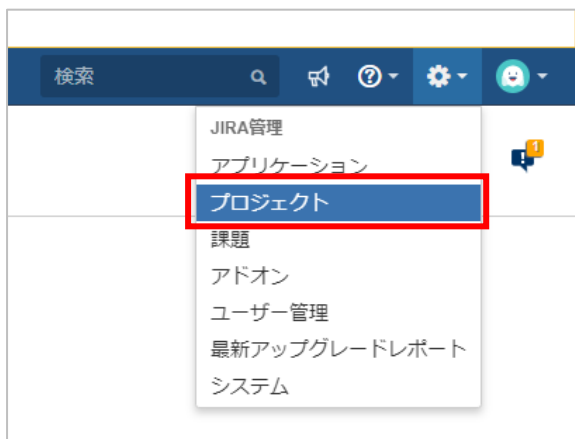
タイトル	QFJIRA
モード	非公開
認証の最大試行回数	3
サインアップ時に CAPTCHA を使用	オフ
ベース URL	http://jira-eval.vtsuite.net:8080
メールの差出人	\${fullname} (JIRA)

1.2.3. 新規チケット作成画面の URL を設定する

新規チケット作成画面の URL を設定すると、テストサイクルでテストを実行中に右クリックから簡単にチケットへの起票を行うことができます。

(1) JIRA の URL の後に「/secure/CreateIssueDetails!init.jspa?pid=**yyy**&issuetype=**zzz**」を追加します。PID と issuetype は以下手順で設定します。

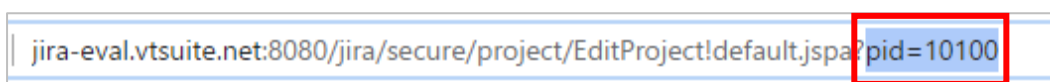
(2) プロジェクト設定を開きます。



(3) 詳細情報を選択します。



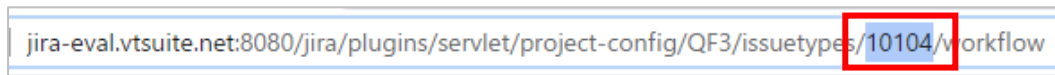
(4) 詳細情報画面の URL の最後に記載されている PID を新規チケット作成画面の URL に設定します。



(5) プロジェクト設定画面で課題タイプを選択します。



(6) 課題タイプの画面 URL に記載されている issuetype を新規チケット作成画面の URL に設定します。



1.2.4. JQL を設定する

JQL は「JIRA Query Language」の略で、JIRA 専用のクエリ言語を指します。取得するプロジェクトや課題のタイプなどを絞り込むために JQL を設定する必要があります。未設定の場合は JIRA に登録されている全てのプロジェクト、課題が対象となります。



例) 特定のプロジェクトを対象とする場合は `project="xxxxx"`

例) 特定の課題タイプを対象とする場合は `issueType = "xxx"`

具体的な入力値について、プロジェクトを対象とする場合は JIRA に登録しているプロジェク

トキーを、課題タイプを対象とする場合は、先の手順で確認した issuetype の値を確認してください。

1.2.5. バグやクローズの文字列を設定する

バグの OPEN・CLOSE を集計する際に、どの課題タイプをバグとみなすか、どのステータスで完了とするかを設定することができます。

バグとみなすタイプ文字列に指定した課題タイプがレポート画面に表示される対象の課題となります。

バグとみなすタイプ文字列

例：バグ,不具合

●カンマ区切りで複数の文字列を指定することができます

クローズとみなすステータス文字列

例：完了,修正済み

●カンマ区切りで複数の文字列を指定することができます

(1) JIRA の管理メニューから課題を選択します。



(2) 課題タイプを開きます。バグとみなすタイプ文字列はこの課題タイプの名前を指定します。

管理 JIRA 管理の検索

アプリケーション プロジェクト **課題** アドオン ユーザー管理 最新アップグレードレポート システム

課題タイプ

課題タイプを追加 ⓘ

名前	タイプ	関連スキーム	アクション
Epic JIRA Softwareにより作成 - 編集 - 削除しないでください。分類の必要がある大きなユーザー ストーリー向けの課題タイプです。	Standard	<ul style="list-style-type: none"> Default Issue Type Scheme QF1: ソフトウェア開発 課題タイプスキーム QF3: ソフトウェア開発 課題タイプスキーム QF4: ソフトウェア開発 課題タイプスキーム QF5: ソフトウェア開発 課題タイプスキーム SC1: スクラム課題タイプスキーム 	編集 削除 翻訳
Story JIRA Softwareにより作成 - 編集 - 削除しないでください。ユーザーのストーリーの課題タイプです。	Standard	<ul style="list-style-type: none"> Default Issue Type Scheme SC1: スクラム課題タイプスキーム 	編集 削除 翻訳
ストーリー JIRA Softwareにより作成 - 編集 - 削除しないでください。ユーザーのストーリーの課題タイプです。	Standard	<ul style="list-style-type: none"> SC1: スクラム課題タイプスキーム 	編集 削除 翻訳
<input checked="" type="checkbox"/> タスク 行うべきタスクを表します。	Standard	<ul style="list-style-type: none"> QF1: ソフトウェア開発 課題タイプスキーム QF2: タスク管理 課題タイプスキーム QF3: ソフトウェア開発 課題タイプスキーム QF4: ソフトウェア開発 課題タイプスキーム QF5: ソフトウェア開発 課題タイプスキーム SC1: スクラム課題タイプスキーム 	編集 削除 翻訳
<input checked="" type="checkbox"/> バグ 製品の機能を損なったり障害する問題を表します。	Standard	<ul style="list-style-type: none"> QF1: ソフトウェア開発 課題タイプスキーム QF3: ソフトウェア開発 課題タイプスキーム QF4: ソフトウェア開発 課題タイプスキーム QF5: ソフトウェア開発 課題タイプスキーム SC1: スクラム課題タイプスキーム 	編集 削除 翻訳
<input checked="" type="checkbox"/> 改善 既存の機能またはタスクに対する改善事項を表します。	Standard	<ul style="list-style-type: none"> QF1: ソフトウェア開発 課題タイプスキーム QF3: ソフトウェア開発 課題タイプスキーム QF4: ソフトウェア開発 課題タイプスキーム QF5: ソフトウェア開発 課題タイプスキーム 	編集 削除 翻訳

- (3) 課題タイプメニューからステータスを開きます。クローズとみなすステータス文字列はこのステータスの名前を指定します。

管理 JIRA 管理の検索

アプリケーション プロジェクト 課題 アドオン ユーザー管理 最新アップグレードレポート システム

課題タイプ
課題タイプ
課題タイプスキーム
サブタスク

ワークフロー
ワークフロー
ワークフロースキーム

画面
画面
画面スキーム
課題タイプ画面スキーム

フィールド
カスタムフィールド
フィールド構成
フィールド構成スキーム

課題の機能
時間のトラッキング
課題のリンク

課題の属性
ステータス
解決状況

ステータス

名前

オープン
課題がオープンになっており、担当者が作業を開始できる状態を表します。

進行中
この課題は担当者によって現在作業が進められていることを表します。

再オープン
課題が一度解決されたが解決に間違いがあったと見なされたことを表します。ここから課題を割り当て済みにするか解決済みに設定できます。

解決済み
解決され、報告者からの確認を待っている状態を表します。ここから課題を再オープンにするかワークフローに設定できます。

クローズ
課題の検討が終了し、解決方法が正しいことを表します。クローズした課題は再オープンすることができます。

To Do

In Review

完了

ステータスを追加 ステータスの翻訳

カテゴリ	ワークフロー	順序	アクション
To Do	2件の関連ワークフロー	↓	編集
進行中	7件の関連ワークフロー	↑ ↓	編集
To Do	2件の関連ワークフロー	↑ ↓	編集
完了	2件の関連ワークフロー	↑ ↓	編集
完了	2件の関連ワークフロー	↑ ↓	編集
To Do	6件の関連ワークフロー	↑ ↓	編集
進行中	4件の関連ワークフロー	↑ ↓	編集
完了	6件の関連ワークフロー	↑	編集

※課題が一部しか表示されない場合は `jira.search.views.default.max` の値を確認してください

※JIRA との接続に失敗する場合は以下の項目を確認してください

1. ログインに失敗する場合は JIRA で直接ログインをした後に連携設定を試みてください
(JIRA のログインを複数回失敗すると CAPTCHA 認証が必要となります)
2. プロジェクト設定権限があることを確認してください
3. JIRA の認証設定が BASIC 認証となっていることを確認してください

1.3. JIRA クラウド型と連携する

1.3.1. ユーザ名とパスワードを入力する

クラウド型をご利用の方は API トークンの作成を行い、メールアドレスとパスワードを入力し

てください。

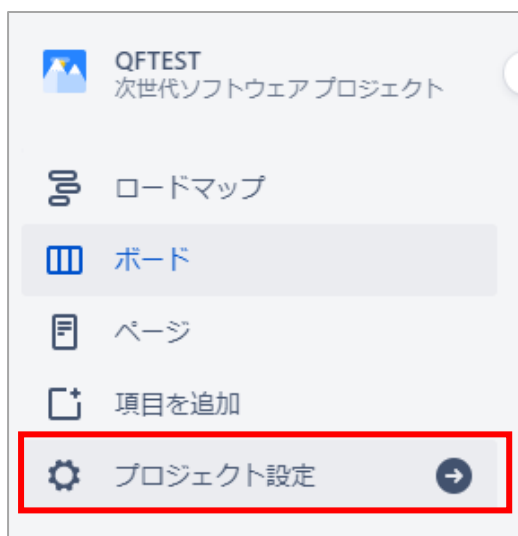
- (1) <https://ja.confluence.atlassian.com/cloud/api-tokens-938839638.html> を参考に API トークンの作成を行ってください。
- (2) ユーザ名/パスワードを以下に設定してください。
ユーザ名 : メールアドレス
パスワード : API トークン

1.3.2. 新規チケット作成画面の URL を設定する

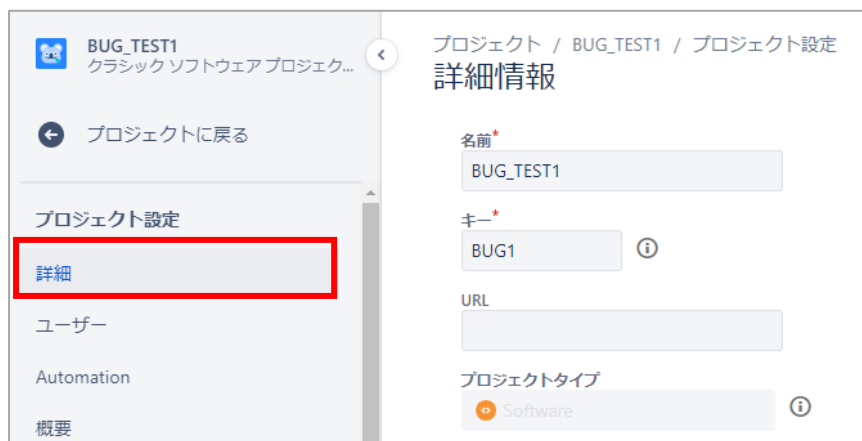
新規チケット作成画面の URL を設定すると、テストサイクルでテストを実行中に右クリックから簡単にチケットへの起票を行うことができます。

クラシックプロジェクトをご利用の場合

- (1) 「クラシックプロジェクト」をご利用の場合、JIRA の URL の後に「/secure/CreateIssueDetails!init.jspa?pid=yyy&issuetype=zzz」を追加します。PID と issuetype は以下手順で設定します。
- (2) プロジェクト設定を開きます。



(3) 詳細情報を選択します。



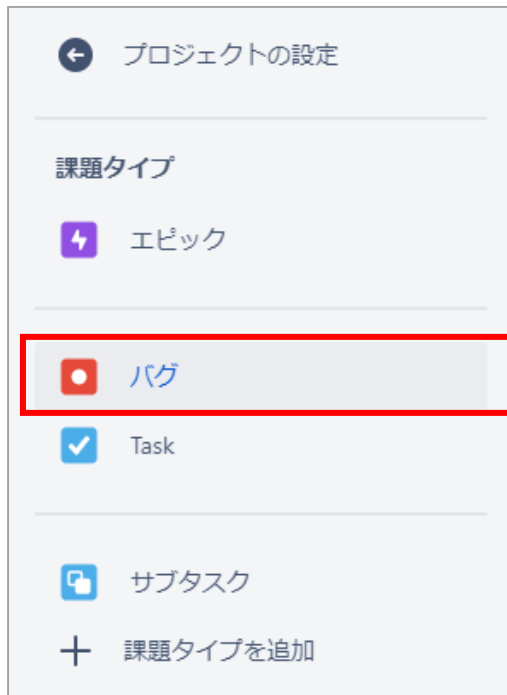
(4) 詳細情報画面の URL の最後に記載されている PID を新規チケット作成画面の URL に設定します。

`/secure/project/EditProject!default.jspa?pid=10000`

(5) プロジェクト設定画面で課題タイプを選択します。



(6) チケット新規作成時にデフォルトにしたい課題タイプを選択します。



- (7) 課題タイプの画面 URL に記載されている issuetype を新規チケット作成画面の URL に設定します。

`/jira/software/projects/QT/settings/issuetypes/10010`

次世代プロジェクトをご利用の場合

次世代ソフトウェアプロジェクトでは、RestAPI から PID をご参照ください。手順は以下の通りです。

- (1) 「次世代プロジェクト」をご利用の場合、JIRA の URL の後に「`/secure/CreateIssueDetails!init.jsps?pid=yyy&issuetype=zzz`」を追加します。PID と issuetype は以下手順で設定します。
- (2) ブラウザに以下の URL を入力してください。JSON の「id」の項目が PID です。
`https://[JIRA の URL]/rest/api/2/project/[プロジェクトキー]`

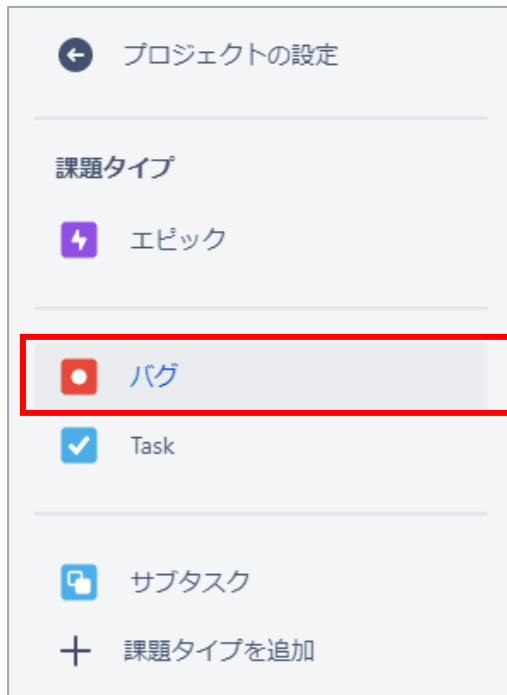
(3) プロジェクト設定画面を開きます。



(4) プロジェクト設定画面で課題タイプを選択します。



(5) チケット新規作成時にデフォルトにしたい課題タイプを選択します。



- (6) 課題タイプの画面 URL に記載されている issuetype を新規チケット作成画面の URL に設定します。

`/jira/software/projects/QT/settings/issuetypes/10010`

1.3.3. JQL を設定する

JQL は「JIRA Query Language」の略で、JIRA 専用のクエリ言語を指します。取得するプロジェクトや課題のタイプなどを絞り込むために JQL を設定する必要があります。未設定の場合は JIRA に登録されている全てのプロジェクト、課題が対象となります。

JQL

例: `project="QF1"`

ⓘ バグ情報の取得に利用します。事前にすべての Issue が「新しい順で」取得できることをご確認ください

例) 特定のプロジェクトを対象とする場合は `project="xxxxx"`

例) 特定の課題タイプを対象とする場合は `issueType = "xxx"`

具体的な入力値について、プロジェクトを対象とする場合は JIRA に登録しているプロジェクトキーを、課題タイプを対象とする場合は、先の手順で確認した issuetype の値を確認してくだ

さい。

1.3.4. バグやクローズの文字列を設定する

バグの OPEN・CLOSE を集計する際に、どの課題タイプをバグとみなすか、どのステータスで完了とするかを設定することができます。

バグとみなすタイプ文字列に指定した課題タイプがレポート画面に表示される対象の課題となります。

<p>バグとみなすタイプ文字列</p> <p>例：バグ,不具合</p> <p>●カンマ区切りで複数の文字列を指定することができます</p>
<p>クローズとみなすステータス文字列</p> <p>例：完了,修正済み</p> <p>●カンマ区切りで複数の文字列を指定することができます</p>

第2章 よくある質問と回答

2.1. Q: BTS 疎通ができているか確認したい

BTS の疎通確認は疎通確認画面より行っていただけます。確認手順は以下の通りです。

- (1) テストフェーズ一覧のテストフェーズ名下部にある「Redmine（または JIRA）との疎通確認」をクリックします。



テストフェーズ名 ▼	フェーズ開始日
テストサンプル_画面遷移テスト 📅 2018/08/30 ~ 2018/09/30 🔗 設定	2018/08/30
サンプルフェーズ東京 📅 2017/02/21 ~ 2017/03/22 🔗 設定 <input checked="" type="checkbox"/> Redmineとの疎通確認	2017/02/21
βリリース向けフル試験 📅 2017/02/21 ~ 2017/03/22 🔗 設定	2017/02/21

- (2) 疎通確認画面で「ベース URL の疎通確認」の項目が「○」となっていれば疎通に成功しています。「×」となっている場合は設定を見直していただく必要がございます。

Redmineとの疎通確認

疎通確認

チェック内容	結果
ベースURLの疎通確認	<input type="radio"/>
バグ曲線、グラフデータ取得用URLの疎通確認	<input type="radio"/>
最近のインシデント取得用URL	<input type="radio"/>

ステータス確認

ステータス	CLOSEと見なすか
New	X
In Progress	X
Resolved	X
Feedback	X
Closed	<input type="radio"/>
Rejected	<input type="radio"/>
完了	<input type="radio"/>

2.2. Q: BTS 疎通ができない

BTS に接続する際に外部のシステムである QualityForward からのアクセスが弾かれてしまう場合がございます。その場合は QualityForward の IP アドレス ([第 1 章参照](#)) からのアクセスを許可していただく必要がございます。

本設定は Redmine のシステム管理者権限を持っているユーザ様にご対応いただく必要がございます。

2.3. Q: BTS の件数がグラフに反映されない

フェーズ毎チャート画面の右上にある設定から「Redmine (または JIRA) と同期する」ボタ

ンを押してください。※BTS の情報が自動で反映されるタイミングは1日3回（8時・12時・18時）です。



2.4. Q: BTS の Open /Close の件数がレポートに反映されるのはいつですか？

1日3回（8時・12時・18時）です。任意のタイミングで更新したい場合は、設定から「Redmine（またはJIRA）と同期する」ボタンを押すことでレポートに反映されます。

2.5. Q: 過去の BTS の Open/Close 数を修正したい

フェーズ毎チャート画面右上にある設定の「バグ情報のアップロード」から修正が可能です。



2.6. Q: JIRA のクラウド型を使用したい

以下の手順に従い設定を行ってください。

- (1) [こちら](#)を参考に API トークンの作成を行ってください。
- (2) ユーザ名/パスワードを以下のように設定してください。
 ユーザ名 : メールアドレス
 パスワード : API トークン

2.7. Q: Redmine で BTS 疎通ができない

Redmine の設定が有効になっていない可能性があります。

- (1) 「管理」 → 「設定」 → 「API」 タブを開きます。

- (2) 「REST による Web サービスを有効にする」にチェックを入れます。



設定

全般 表示 認証 API プロジェクト チケットトラッキング ファイル

RESTによるWebサービスを有効にする

JSONPを有効にする

保存

- (3) 「保存」ボタンを押します。

2.8. Q: BTS で集計されるチケット範囲を絞りたい

集計されるチケットの範囲を絞るには、Redmine 側で絞り込みを行ったものを指定していただく必要がございます。Redmine での絞り込み手順は以下の通りです。

- (1) Redmine の個人設定から API キーを取得します。
- (2) Redmine の対象のプロジェクト開き、URL を取得します。
例) <https://xxx.xxxx/projects/xxxx/>
- (3) 手順(2)で取得した URL に「issues.json?key=API キー」を追加します。例)
<https://xxx.xxxx/projects/xxxx/issues.json?key=xxxx>
- (4) Redmine で対象チケットの絞り込みを行います。バグ一覧を取得するために「すべて」の「バグ」を対象にし、適用ボタンを押します。

チケット

▼ フィルタ

ステータス すべて ▼

トラッカー 等しい ▼ Bug ▼ □

▼ オプション

適用

※ここでは「すべて」の「バグ」を対象にしていますが、フィルタ条件はプロジェクト方針に合わせて自由に設定していただけます。

- (5) すべてのバグの一覧が表示されたら保存ボタンを押します。

チケット

▼ フィルタ

ステータス すべて ▼

トラッカー 等しい ▼ Bug ▼ □

▼ オプション

適用

- (6) 新しいクエリに名前を付けて保存します。

新しいクエリ

名称

表示 自分のみ
 すべてのユーザー
 次のロールのみ:
 Manager
 Developer
 Reporter

全プロジェクト向け

オプション

デフォルトの項目

グループ条件 ▼

表示 説明

合計 予定工数 作業時間

フィルタ

ステータス すべて ▼

トラッカー 等しい ▼ Bug ▼ □

ソート条件

1: ▼ ▼

2: ▼ ▼

3: ▼ ▼

- (7) チケット一覧右側のカスタムクエリ一覧に手順(6)で作成したクエリが表示されます。作成したクエリ名をクリックします。



- (8) URL の最後にクエリ ID が表示されるので、「query_id=xx」をコピーします。

/issues?query_id=25

- (9) 手順(3)までで作成した URL の最後に手順(8)の「&query_id=xx」を入力します。
例)https://xxx.xxxx/projects/xxxx/issues.json?key=xxxx&query_id=xx

- (10) 手順(9)でできた URL をブラウザのアドレス欄に直接入力します。json の取得が確認できたら URL を登録欄に入力し、登録ボタンを押します。

2.9. Q: チケットを CLOSE にしたのに欠陥 OPEN 数が変わらない

欠陥 OPEN 数はチケットの総数を表しているため、グラフが減ることはありません。