

# BTS 連携マニュアル

VERISERVE

最終更新日:2025/05/28

# 目次

第1章 [	3TS 連携時の注意点
1.1.1	
1.1.2	. BTS 別確認事項
第2章 日	3TS 連携を行う4
2.1. I	Redmine と連携する4
2.1.1	. Redmine のベース URL を設定する5
2.1.2	. バグ曲線、グラフデータ取得用 URL を設定する6
2.1.3	. 最近のインシデント取得用 URL を設定する9
2.1.4	. 新規チケット作成画面の URL9
2.2.	IIRA サーバー型と連携する9
2.2.1	. ユーザ名とパスワードを入力する10
2.2.2	. URL・コンテキストパスを設定する11
2.2.3	. バグ曲線、グラフデータ取得用 JQL を設定する12
2.2.4	. 最近のインシデント取得用 JQL を設定する13
2.2.5	. 新規チケット作成画面の URL を設定する13
2.3.	IIRA クラウド型と連携する15
2.3.1	. ユーザ名とパスワードを入力する16
2.3.2	. バグ曲線、グラフデータ取得用 JQL を設定する17
2.3.3	. 最近のインシデント取得用 JQL を設定する17
2.3.4	. 新規チケット作成画面の URL を設定する18
2.4. I	Backlog と連携する22
2.4.1	. 基本設定を入力する
2.4.2	. 連携する BTS の追加設定を行う 27
第3章。	よくある質問と回答
3.1. (	Q: BTS 疎通ができているか確認したい
3.2. (	Q: BTS 疎通ができない
3.3. (	Q: BTS の件数がグラフに反映されない31

3.4.	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?	. 31
3.5.	Q:過去の BTS の Open/Close 数を修正したい	. 32
3.6.	Q: JIRA のクラウド型を使用したい	. 32
3.7.	Q: Redmine で BTS 疎通ができない	. 33
3.8.	Q: BTS で集計されるチケット範囲を絞りたい	. 33
3.9.	Q: チケットを CLOSE にしたのに欠陥 OPEN 数が変わらない	. 36

# 第1章 BTS 連携時の注意点

BTS 連携をご利用の際、以下の確認が必要です。

# 1.1.1. 共通確認事項

REST API 設定	有効化されていることを確認してください
IP 制限	アクセス制限が有効な場合、以下 IP アドレスからのアクセスを許可してく
	ださい。
	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
	54.95.210.141

# 1.1.2. BTS 別確認事項

JIRA Cloud 版		標準サポート
JIRA Server 版	7.0 以上	AWS 上の QualityForward クラウド版から、ホスティン
		グしている環境への HTTP 通信を穴開けする必要がありま
		す。
Redmine	2.0 以上	AWS 上の QualityForward クラウド版から、ホスティン
		グしている環境への HTTP 通信を穴開けする必要がありま
		す。

# 第2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。

# 2.1. Redmine と連携する

テストフェーズ一覧のテストフェーズ設定から、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。

テストフェーズ一覧
▶ アクティブ 6 🖬 アーカイブ 🛛
名前で検索
テストフェーズ名▲
0525_1 ☆ 2021/05/25 ~ 2021/06/25
<b>0525_2</b> 首 2021/05/25 ~ 2021/06/25 <i>2</i> 設定

## 連携するBTS BTS連携に関する設定はこちらの資料をご確認下さい。 BTS なし なし Redmine JIRA 連携するBTS 🧡 BTS連携とは BTS連携に関する設定はこちらの資料をご確認下さい。 BTS ~ Redmine <u>必須</u> 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 M バグ曲線 パイチャート取得田URI

221 ハク 田稼、 ハイ チャート IX 特用 UKL
例:https://xxx/projects/test/issues.json?key=yyyy&query_id=x
●バグ情報とチャート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
連携するBTSの追加設定
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
●レポート画面で最近のインシデントにバグ曲線と別の情報を表示したい場合に設定してください。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
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

更新する

# 2.1.1. 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://xxxxxxxxxxxx/] 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。

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。

Atomアクセスキー					
Atomアクセスキーは9ヶ月前に作成されました (リセット)					
APIアクセスキー					
表示					
APIアクセスキーは	。 約1年前に作成されました (リセット)				

# 2.1.2. バグ曲線、グラフデータ取得用 URL を設定する

バグ曲線とパイチャート表示用の URL の入力を行います。本手順で作成するクエリ設定で、取

得するチケットを絞込むことができます。

```
        Xグ曲線、パイチャート取得用URL

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

        Sバグ情報とチャート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.xxx/projects/xxx/

- (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.xxx/projects/xxxx/issues.json?key=xxxx
- (4) 取得する情報のフィルタ条件を設定し、対象チケットの絞込みを行います。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。

概要	活動	ロードマップ	<i>ች</i> ታット	新しいま	チケット	ガントチャート	カレンダ	- ב-ג	大書	Wiki	ファイル	設定
チケッ	אר											
- v 74)	1/9											
✓ 75	テータス		-	すべて	•							
💌 ht	ラッカー		ŧ	等しい	۲	Bug		Ŧ				
– ▶ オプ:	ションー											
✔ 適用 (	<b>)</b> クリ:	ア 📙 保存										

※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。

概要	活動	ロードマップ	<i>ች</i> ታット	新しいチ	ታット	ガントチャート	カレンダー	ג-ב	文書	Wiki	ファイル	設定
チケ	ישר											
7-	าルター											
	マテータス		-	すべて	•							
	ヽラッカー		1	等しい	•	Bug		•				
- ⊩ オ:	プションー											
🖌 適用	1 🦻 クリ	ア 📙 保存										

(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 <ul> <li>自分のみ</li> <li>すべてのユーザー</li> <li>次のロールのみ:</li> <li>Manager</li> <li>Developer</li> <li>Reporter</li> </ul>
全プロジェクト向け 📄
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
7111/9
<ul> <li></li></ul>
ソート条件       1:     ▼       2:     ▼       3:     ▼
保存

(7) チケット一覧右側のカスタムクエリー覧に手順(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 サフリー
カレンダー
カントチャート インポート
カスタムクエリ
全てのパグ

- (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<sup>1</sup>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。

### 2.1.3. 最近のインシデント取得用 URL を設定する

連携した BTS の「最近のインシデント一覧」を表示するため、指定された範囲のチケット情報 を取得します。未設定の場合は「バグ曲線、グラフデータ」に設定した URL から情報を取得しま

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
●レポート画面で最近のインシデントにバグ曲線と別の情報を表示したい場合に設定してください。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 2.1.2 と同様に行います。

※最近のインシデント取得用 URL はバグ曲線やチャートと別の情報を表示させたい場合に指定 してください。

### 2.1.4. 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

### 2.2. JIRA サーバー型と連携する

テストフェーズ一覧のテストフェーズ設定から、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。

テストフェーズ一覧
▶ アクティブ 6 🖬 アーカイブ 0
名前で検索
テストフェーズ名▲
0525_1
☐ 2021/05/25 ~ 2021/06/25
₫ 設定
0525_2
☐ 2021/05/25 ~ 2021/06/25
∅ 設定
連携するBTS
BTS連携に関する設定はこちらの資料をご確認下さい。
BTS
なし 🗸
なし
Redmine
JIRA

# 2.2.1. 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。

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。
※約 ユーザ名
ユーザ名
<u>参照</u> パスワード
パスワード

# 2.2.2. 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。

Magna Jirao Url
https://xxx.jp/
移動 バグ曲線、パイチャート取得用JQL
例:project="QF1"
●バグ情報とチャート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
連携するBTSの追加設定
コンテキストパス
例:/jira
❶JIRA側で設定してる場合にのみ入力してください

#### (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。

ŸJIRA ダッシュボード・ プロ	]ジェクト・ 課題・ ボード・ 作成	
管理 ۹. 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
ー <b>般設定</b> 管理ツールをさらに検索	設定	
	一般設定	
システム情報	タイトル	QFJIRA
インストゥルメンテーショ	Ξ-ド	非公開
ンークが一つ所有	認証の最大試行回数	3
アータベース監視	サインアップ時に CAPTCHA を使用	オフ
ログとプロファイルの作成	ベース URL	http://jira-eval.vtsuite.net:8080
スケジューラー情報	メールの差出人	\${fuliname} (JIRA)
トラブルシューティングと サポートツール	概要	
監査ログ	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。

設定	
一般設定	
タイトル	QFJIRA
オーチ	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	

# 2.2.3. バグ曲線、グラフデータ取得用 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。

❷测 バグ曲線、パイチャート取得用JQL
例:project="QF1"
❶バグ情報とチャート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="プロジェクトキー"

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

※OPEN/CLOSE 情報は JIRA 側の設定に依存します。

## 2.2.4. 最近のインシデント取得用 JQL を設定する

連携した BTS の「最近のインシデント一覧」を表示するため、指定された範囲のチケット情報 を取得します。未設定の場合は「バグ曲線、グラフデータ」に設定した JQL から情報を取得しま

す。

連携するBTSの追加設定
コンテキストパス
例:/jira
● JIRA側で設定してる場合にのみ入力してください
最新のインシデント用JQL
例:project = QF AND issuetype = バグ
●「パグ曲線、グラフデータ取得用JQL」と異なる情報を表示したい場合に入力してください

※URLの取得は手順 2.2.3 と同様に行います。

※最近のインシデント取得用 JQL はバグ曲線やチャートと別の情報を表示させたい場合に指定 してください。

# 2.2.5. 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。

- 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。

プロジェクト設定	
要約 詳細情報 プロジェクトを再インデックス化 プロジェクトを削除 課題タイプ 課題タイプ エピック ・サブタスク ・タスク ・バグ ・び当 ・び当 ・新機能	詳細情報          名前*       テストプロジェクト1         キー*       QF3         URL       (i)         プロジェクトタイプ*       ③ Software         プロジェクトカテゴ       None         リー       アバター*         ジョのの選択

(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。 jira-eval.vtsuite.net:8080/jira/secure/project/EditProject!default.jspa?pid=10100

(5) プロジェクト設定画面で課題タイプを選択します。

プロジェクト設定	
要約	
詳細情報	
プロジェクトを再インデッ クス化	
プロジェクトを削除	
課題タイプ	
- エピック	
- サブタスク	
- タスク	
パグ	
- 改善	
新機能	

(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-eval.vtsuite.net:8080/jira/plugins/servlet/project-config/QF3/issuetypes/10104/vorkflow

※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

### 2.3. JIRA クラウド型と連携する

テストフェーズ一覧のテストフェーズ設定から、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。



# 2.3.1. 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(API ト ークン)を入力してください。

 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

# 2.3.2. バグ曲線、グラフデータ取得用 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"
のパ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="プロジェクトキー"

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

※OPEN/CLOSE 情報は JIRA 側の設定に依存します。

### 2.3.3. 最近のインシデント取得用 JQL を設定する

連携した BTS の「最近のインシデント一覧」を表示するため、指定された範囲のチケット情報 を取得します。未設定の場合は「バグ曲線、グラフデータ」に設定した JQL から情報を取得しま す。

連携するBTSの追加設定
コンテキストパス
例:/jira
❶JIRA側で設定してる場合にのみ入力してください
最新のインシデント用JQL
例:project = QF AND issuetype = バグ
❶「バグ曲線、グラフデータ取得用JQL」と異なる情報を表示したい場合に入力してください

※URLの取得は手順 2.3.2 と同様に行います。

※最近のインシデント取得用 JQL はバグ曲線やチャートと別の情報を表示させたい場合に指定 してください。

## 2.3.4. 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) 「クラッ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。

BUG_TEST1 クラシックソフトウェアプロジェク ・	プロジェクト / BUG_TEST1 / プロジェクト設定 詳細情報
<ul><li>プロジェクトに戻る</li></ul>	名前 <sup>*</sup> BUG_TEST1
プロジェクト設定	‡-*
詳細	BUG1
ユーザー	URL
Automation	プロジェクトタイプ
概要	<ul> <li>Software</li> <li>Image: Image of the second sec</li></ul>

(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。

プロジェクトの設定	
詳細	
アクセス	
課題タイプ	Ð
通知	
機能	
アプリ	

(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。

G	プロジェクトの設定
課題	タイプ
5	エピック
	バグ
<ul> <li>Image: A set of the set of the</li></ul>	Task
6	サブタスク
+	課題タイプを追加

(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<mark>/</mark>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.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) ブラウザに以下の URL を入力してください。JSON の一番上の「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。

プロジェクトの設定	
詳細	
アクセス	
課題タイプ	Ð
通知	
機能	
アプリ	

(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。

G	プロジェクトの設定
課題	タイプ
5	エピック
	バグ
<ul> <li>Image: A start of the start of</li></ul>	Task
6	サブタスク
+	課題タイプを追加

(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

# 2.4. Backlog と連携する

テストフェーズ一覧のテストフェーズ設定から、連携する BTS で「Backlog」を選択すると Backlog に連携 するための設定項目が表示されます。



力してください。

なし

JIRA

Redmine

Backlog

連携するBTS 🤚 BTS連携とは
BTS連携に関する設定はこちらの資料をご確認下さい。
Backlog ~
83 スペースID
BacklogのスペースIDを入力してください。
◎別 プロジェクトキー
Backlogのプロジェクトキーを入力してください。
BacklogのAPIキーを入力してください。
連携するBTSの追加設定
バグ曲線、パイチャート取得用URL
https://your_space_id.backlog.com/api/v2/issues?apiKey=your_api_key&issueTypeId[]=your_issue_type_id&count=100&projectId[]=your_project_id
●バグ情報とチャート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。このURLの指定がない場合、対象プロジェクトで「バグ」になっている課題を取 得します。
最近のインシデント取得用URL
https://your_space_id.backlog.com/api/v2/issues?apiKey=your_api_key&issueTypeId[]=your_issue_type_id&count=100&projectId[]=your_project_id&sort=upda
●レポート画面で最近のインシデントにバグ曲線と別の情報を表示したい場合に設定してください。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。
更新する

# 2.4.1. 基本設定を入力する

#### スペース ID を指定する

Backlog のホーム画面の URL からスペース ID をご確認いただき、スペース ID を入力します。

例)https://xxxxxx.backlog.com の xxxxxx 部分がスペース ID です。



#### プロジェクトキーを指定する

Backlog のプロジェクトホーム画面を開きます。サイドのメニューバーで「ホーム」を選択した画面がホーム 画面です。



ホーム画面を開いた際の URL の末尾を確認し、プロジェクトキー欄に入力します。プロジェクトキーはプロジェクト作成時に指定したキーです。

例)<u>https://xxxxx.backlog.com/projects/your\_project\_key</u>のyour\_project\_key部分がプロジェクトキーです。



#### API キーを指定する

Backlog での API キーの取得方法は以下の通りです。

(1)Backlog 上部の個人アイコンをクリックし、個人設定をクリックします。



(2)個人設定メニューから「API」を選択します。

<b>章</b> 個人設定	APIの設定 ⑦
ユーザー情報	新しいAPIキーを発行
メール設定	
プライベートアドレス	XE
SSH 22MB	
アプリ連携	9213
API	

#### (3)任意のメモを入力し、登録ボタンを押します。

<b>章</b> 個人認定	APIの設定 ⑦
ユーザー情報	新しいAPは一を発行
メール設定	
プライベートアドレス	XE
SSH 公開館	
アプリ連携	1738
API	

(4)登録された API キーをコピーし、API キー欄に入力します。

登録されたAPIキー				
API‡—	2Ľ-	×т	登録日	削除
		QualityForward	2025/04/16	×

# 2.4.2. 連携する BTS の追加設定を行う

連携した BTS から情報を取得するためのクエリを作成します。

#### バグ曲線、パイチャート取得用 URL を指定する

次に、バグ曲線とパイチャート取得用の URL を作成します。取得対象とするチケットを絞り込むことができます。URL の作成手順は以下の通りです。

バグ曲線、パイチャート取得用URL https://your\_space\_id.backlog.com/api/v2/issues?apiKey=your\_api\_key&issueTypeId[]=your\_issue\_type\_id&count=100&projectId[]=your\_project\_id ●バグ情報とチャート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。このURLの指定がない場合、対象プロジェクトで「バグ」になっている課題を取 得します。

#### (1) Backlog の対象のプロジェクトを開き、ベース URL を取得します。

例)<u>https://your\_space\_id.backlog.com/</u>

 (2)手順(1)で取得した URL に「api/v2/issues?apiKey=your\_api\_key」を追加し、取得した API キーを入 力します。

例)<u>https://your\_space\_id.backlog.com/api/v2/issues?apiKey=your\_api\_key</u>

- (3) 手順(2) で作成した URL に「<u>&issueTypeId</u>]=your\_issue\_type\_id&projectId[]=your\_project\_id」を追加し ます。
- (4) issueTypeId を取得します。Backlog でプロジェクト設定>種別>絞り込み対象としたい種別を選択し、URL の「issueType.id=」で表示されている数字が issueTypeId です。



(5) projectId を取得します。Backlog でプロジェクト設定を開きます。URL の「project.id=」の箇所に表示さ れている数字が projectId です。



#### 最終的な URL のフォーマットは以下の通りです。

https://your\_space\_id.backlog.com/api/v2/issues?apiKey=your\_api\_key&issueTypeId[]=your\_issue\_typ e\_id&projectId[]=your\_project\_id

#### 最近のインシデント取得用 URL

連携した BTS の「最近のインシデントー覧」を表示するため、指定された範囲のチケット情報を取得します。未設定の場合は「バグ曲線、パイチャート取得用 URL」に設定した URL から情報を取得します。

最近のインシデント取得用URL
$https://your\_space\_id.backlog.com/api/v2/issues?apiKey=your\_api\_key&issueTypeId[]=your\_issue\_type\_id&count=100&projectId[]=your\_project\_id&sort=updablesetereeset$
●レポート画面で最近のインシデントにバグ曲線と別の情報を表示したい場合に設定してください。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「<u>バグ曲線、パイチャート取得用 URL を指定する</u>」と同様に行います。 ※最近のインシデント取得用 URL はバグ曲線、パイチャートと別の情報を表示させたい場合に指定 してください。

# 第3章 よくある質問と回答

## 3.1. 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。

テストフェーズ一覧	
▶ アクティブ 4 📑 アーカイブ 0	
名前で検索	検索
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	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「O」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							
,							
チェック内容	結果						
ベースURLの疎通確認	0						
バグ曲線、グラフデータ取得用URLの疎通確認	0						
最近のインシデント取得用URL	0						

### 3.2. 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(<u>第1章</u>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。

## 3.3. 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「欠陥実績の更新」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。



3.4. 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、「欠陥実績の更新」ボタンを押すことでレポートに反映されます。

### 3.5. 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。



## 3.6. 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

### 3.7. 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。

### 3.8. 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.xxx/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。

概要 活動 ロードマップ チク	ット 新しいチケット	ガントチャート	カレンダー	ג−ב⊐	文書	Wiki	ファイル	設定
チケット								
– マ フィルター								
🕑 ステータス	すべて							
✔ トラッカー	等しい 「	Bug		T	-			
• พ.ศ. 2937								
✔ 適用 🗊 クリア 🛃 保存								

※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。

概要	活動	ロードマップ	<i>チ</i> ケット	新しいヲ	<mark>የ</mark> ታット	ガントチャー	-ト	カレンダー	ג−ב	文書	Wiki	ファイル	設定
チケ	ット												
7-	าルター												
	マテータス		-	すべて	•								
	<b>ト</b> ラッカー		4	等しい	•	В	ıg		•	÷			
- ⊪ 7: 	プション-	_											
✔ 適用	<b>9</b> クリ	7 📄 保存											

(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 <ul> <li>自分のみ</li> <li>すべてのユーザー</li> <li>次のロールのみ:</li> <li>Manager</li> <li>Developer</li> <li>Reporter</li> </ul>
全プロジェクト向け 🔲
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
フィルター
<ul> <li>✓ ステータス すべて ▼</li> <li>✓ トラッカー 等しい ▼</li> <li>Bug ▼</li> </ul>
ソート条件       1:     ▼       2:     ▼       3:     ▼
保存

(7) チケット一覧右側のカスタムクエリー覧に手順(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
全てのバグ

- (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 ORL を登録欄に入力し、登録ボタンを押します。

# 3.9. 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。