デベロッパー向け

ロボアプリ品質チェックリスト解説

ソフトバンクロボティクス株式会社

Pepperパートナープログラム

目次

1. ロボアプリ品質チェックリスト概要
2. ロボアプリ品質チェックリスト利用方法
3. チェック項目解説（抜粋）
    a. 全分岐網羅試験
    b. 負荷耐久試験
    c. 実環境試験
    d. UX観点レビュー
    e. ソースレビュー（抜粋箇所はありません）
    f. Manifest
    g. その他

1. ロボアプリ品質チェックリスト概要

ロボアプリ品質チェックリストとは

ロボアプリ品質チェックリストとは、ロボアプリの品質を上げるための項目チェックリストです。
本チェックリストを全て満たすことで、ロボアプリの品質を向上させることができます。

また、**2017年7月1日**から、ロボアプリ安全性審査の申請時に本チェックリストを用いたテスト **結果を提出**していただく必要がございます。全てを満たしていることが審査通過の条件となりますので、ご承知おきください。

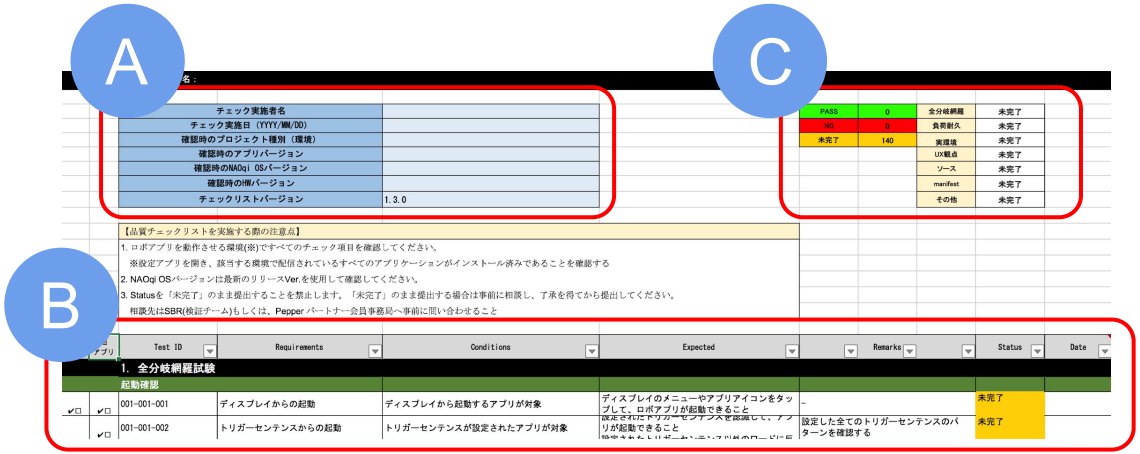
https://developer.softbankrobotics.com/jp-ja/documents/checklist

ダウンロードしてください

2. ロボアプリ品質チェックリスト利用方法

ロボアプリ品質チェックリストの構成

1. ダウンロードしたファイル（チェックリスト）を開いてください。チェックリストの構成は以下のようになっています。



A ロボアプリの基本情報

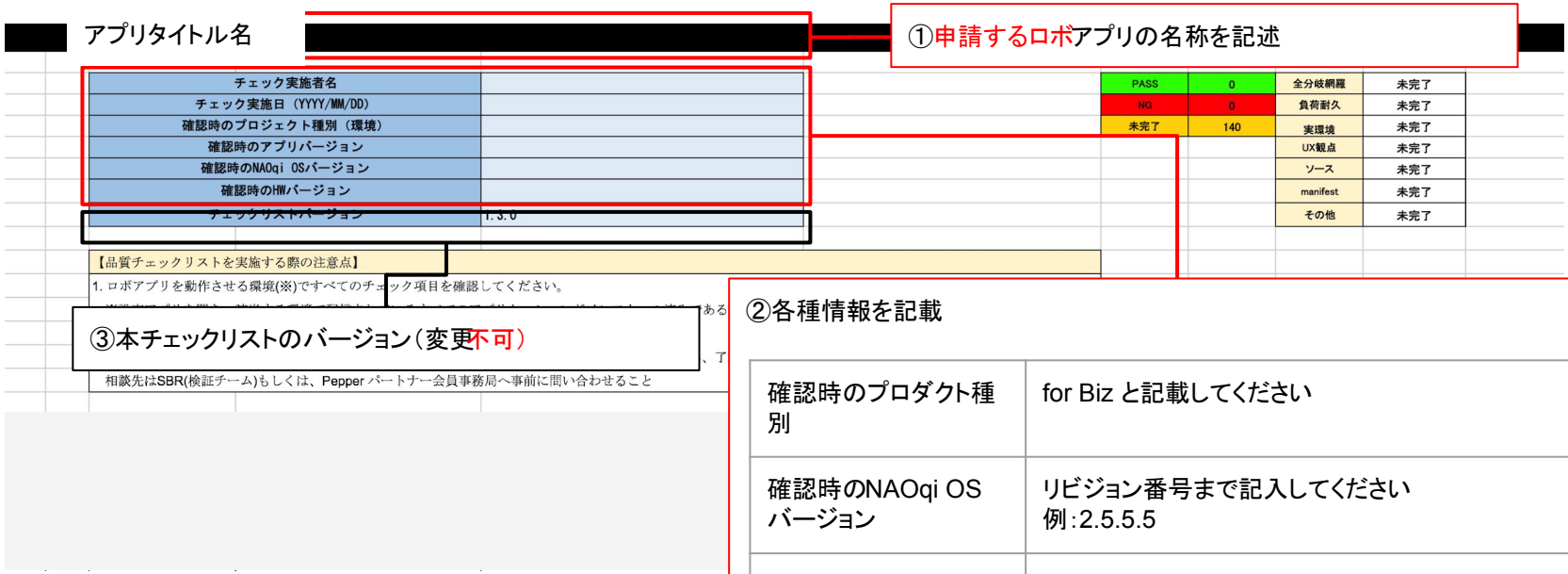
B テスト項目

C 集計結果

2. ロボアプリの基本情報を記載します。
3. 該当するテスト項目で全て Status が PASS となることを確認します。
   全て **PASS になって初めて安全性審査を申請** できます。

ロボアプリ品質チェックリストの使い方 – A

ロボアプリの基本的な情報を記載します



②各種情報を記載

確認時のプロダクト種別	for Biz と記載してください
確認時のNAOqi OS バージョン	リビジョン番号まで記入してください 例：2.5.5.5
確認時のHWバージョン	1.6 と記載してください。 但し、タブレットがアップグレードされているPepperの場合は 1.8a と記載してください。

ロボアプリ品質チェックリストの使い方 – B

チェックリストの項目に従いテストを実施します

アプリタイトル名：

チェック実施者名	
チェック実施日（YYYY/MM/DD）	
確認時のプロジェクト種別（環境）	
確認時のアプリバージョン	
確認時のNAOqi OSバージョン	
確認時のHWバージョン	
チェックリストバージョン	1.3.0

PASS	0	全分岐網羅	未完了
NG	0	負荷耐久	未完了
未完了	140	実環境	未完了
		UX観点	未完了
		ソース	未完了
		manifest	未完了
		その他	未完了

【品質チェックリストを実施する際の注意点】

2. NAOqi OSバージョンは最新のリリースVer.を使用して確認してください。
3. Statusを「未完了」のまま提出することを禁止します。「未完了」のまま提出する場合は事前に相談し、了承を得てから提出してください。
相談先はSBR(検証チーム)もしくは、Pepper パートナー会員事務局へ事前に問い合わせること

マイアプリ	他アプリ	Test ID	Requirements	Conditions	Expected	Remarks	Status	Date
		1. 全分岐網羅試験						
		起動確認						
✔□	✔□	001-001-001	ディスプレイからの起動	ディスプレイから起動するアプリが対象	ディスプレイのメニューやアプリアイコンをタップして、ロボアプリが起動できること	-	未完了	
	✔□	001-001-002	トリガーセンテンスからの起動	トリガーセンテンスが設定されたアプリが対象	設定されたトリガーセンテンスを認識して、アプリが起動できること	設定した全てのトリガーセンテンスのパターンを確認する	未完了	

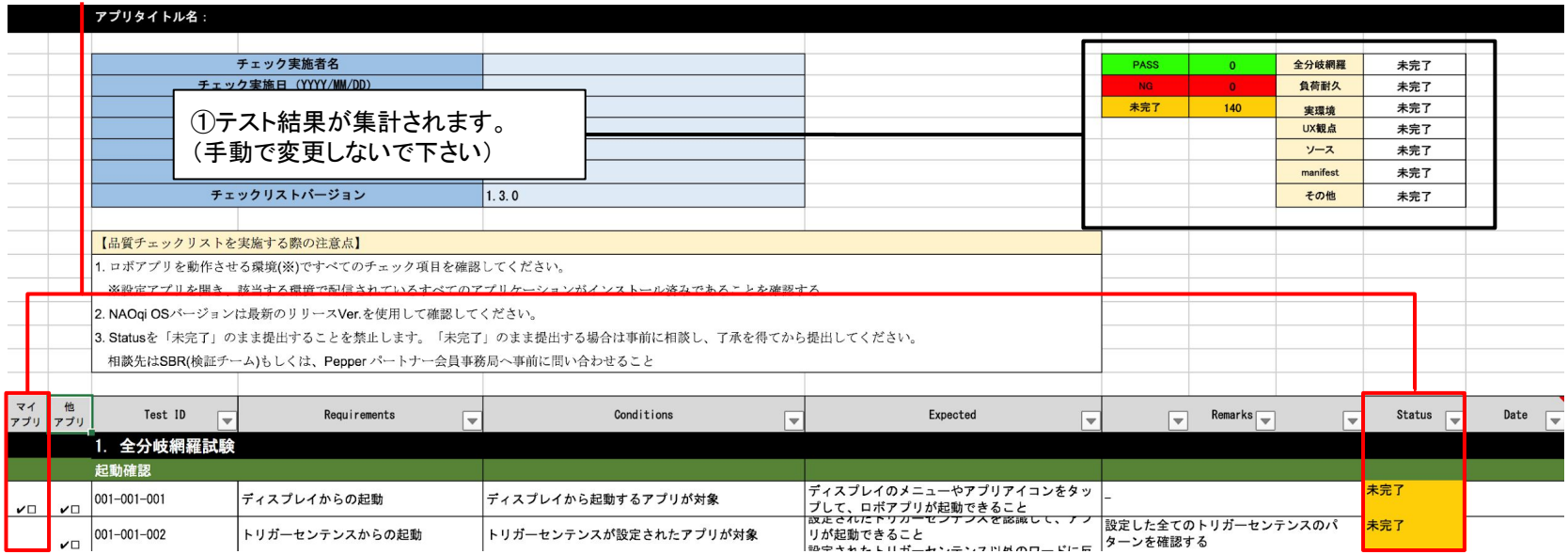
②Pepper for Biz では「マイアプリ」に✔□がある項目が対象となります。

①テスト結果を記入します。

③テスト項目が記述されています。項目の説明については「本書の見方」を参照してください。

ロボアプリ品質チェックリストの使い方 – C

結果が集計されます。全ての該当項目が PASS になっていることを確認してください。



PASS で無い項目がある場合は、全ての項目が PASS となるようにロボアプリを改修してください。

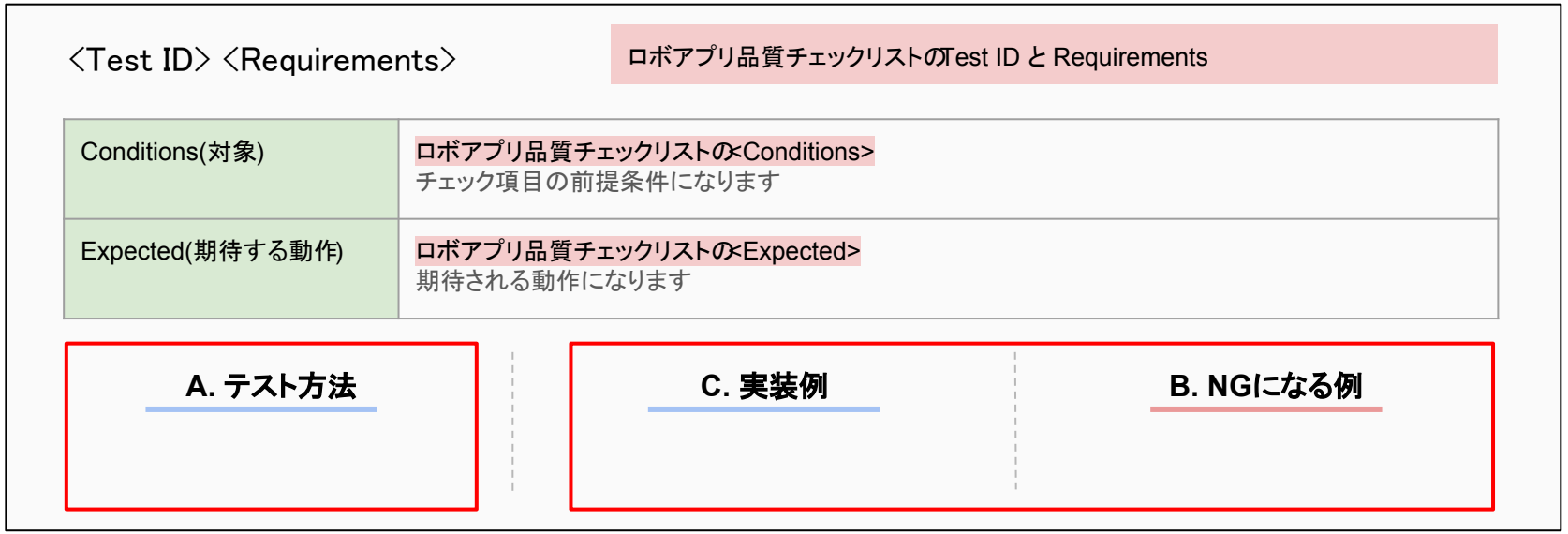
3. チェック項目解説（抜粋）

ロボアプリ品質チェックリストの構成と抜粋箇所

チェック項目は以下の7つのパートから構成されています。
ロボアプリ品質チェックリストの中で特に説明が必要な部分について抜粋して解説をしております。

1. 全分岐網羅試験	001-001-005 アプリ単体での起動
2. 負荷耐久試験	002-001-009 アプリ中断操作後のアプリ起動（複数回）
3. 実環境試験	003-005-003 タブレットと本体が別APに接続
4. UX観点レビュー	004-002-002 ASRの動作 004-005-005 バランス
5. ソースレビュー	本ドキュメントでは触れません
6. Manifest	006-002-009 対象NAOqiのMaxVersion
7. その他	007-001-003 アプリ独自のログ出力有無 007-001-004 データ保存場所

解説の見方と使い方



使い方

1. A[テスト方法]を確認します。
2. A[テスト方法]に従いテストを実施し結果がPASSとなることを確認します。B[NGになる例]を再確認し、NGに該当しないことを確認します。該当した場合はC[実装例]に従いロボアプリを修正します。

001-001-005　アプリ単体での起動

Conditions(対象)	-（全てのロボアプリ）
Expected(期待する動作)	ロボアプリ単体で起動できること 他のロボアプリを呼び出したり、連携していないこと

テスト方法

・ロボアプリが1つのロボアプリで構成されているか確認する。
（ロボアプリが他のロボアプリに依存せず単独で動作可能であること）
・switchFocus APIの呼び出しが無いことを確認する。

実装例

・他のロボアプリの呼び出しを行こなわない。
以下のAPIによる同一ロボアプリ内における他ビヘイビアの呼び出しは許容する。

```
void
ALBehaviorManagerProxy::runBehavior()
```

NGになる例

・他ロボアプリを呼び出している。

```
int ALAutonomousLifeProxy::switchFocus()
```

002-001-004　連続操作

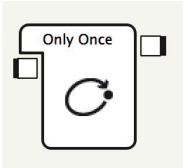
Conditions(対象)	ディスプレイ上にボタンが配置されたアプリが対象
Expected(期待する動作)	アプリ起動中に以下の操作を行い、異常が発生しないこと - 素早く連続的な画面遷移 - ボタン連打 - フリック、画面スクロール - ピンチイン、ピンチアウト

テスト方法

・連打打ち、アプリが落ちるかなどを確認する。危害を加えていない、フリーズやキューイング（音などは多い→Pepper落ちる）などにならないかを確認する。
・連続打ちで次の画面を選択した事になっていないかを確認する。
・フリック、画面スクロールしないかを確認する。

実装例

・[Only Once]ボックスを利用した重複メッセージ送信の抑止を行う。



・応答確定時のディスプレイ操作ロックを実装する。
・ピンチアウト操作のロックを行う。

NGになる例

・意図しないタブレット操作によるアプリケーションのスタックしてしまう。
・画面スクロールやピンチアウト（イン）でのレイアウトが破綻してしまう。

002-001-005　同時押し

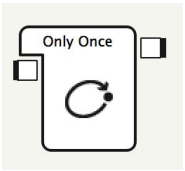
Conditions(対象)	ディスプレイ上にボタンが配置されたアプリが対象
Expected(期待する動作)	アプリ起動中に以下の操作を行い、異常が発生しないこと - 同時押し

テスト方法

・2本指での同時押し、2つのボタンを同時押しの時に、どちらが動くか、アプリが落ちるかなどを確認する。
・危害を加えていない、フリーズやキューイング（音などは多い→Pepper落ちる）などにならないかを確認する。

実装例

・[Only Once]ボックスを利用した重複メッセージ送信の抑止を行う。



・応答確定時のディスプレイ操作ロックを実装する。
・ピンチアウト操作のロックを行う。

NGになる例

・異なるボタンを同時に操作し、それぞれのプログラムが並行して動き、アプリの進行が停止ないし破綻してしまう。

002-001-006　アプリ長時間起動

Conditions(対象)	長時間起動を想定したアプリが対象 ※タイムアウトせずに起動し続けるアプリ
Expected(期待する動作)	アプリ起動した状態のまま1時間以上放置し、モーターホットが発生しないこと

テスト方法

・アプリを起動させ、1時間以上放置させせ不具合が発生しないことを確認する。

実装例

・専用のテストプログラムを開発しモーションに無理がないことを事前に確認する。
・タイムアウトを行わない箇所では、関節を長時間連続で固定するポーズがないようにする。

NGになる例

・アプリを起動し連続稼働を行った場合、モーターホットが発生してしまう。

002-001-007　ユースケースを考慮した稼働負荷

Conditions(対象)	-（全てのロボアプリ）
Expected(期待する動作)	想定範囲の稼働時間で稼働し続け、ユースケースとして想定される以上(混雑時)を考慮して一時的に高負荷を与える、時間を設けた稼働負荷 システム（CPU、Memory）に負荷を掛け過ぎず、アプリ終了後にシステムを解放し、クリーンな状態にしていること

テスト方法

・正常動作以外を操作を行い待機させる。
（例：顔認識の状態で待機させるなど）

・コマンド htopで状態確認を行う。

実装例

・タイムアウトにて制御
※特にユーザの応答を待つタイミング
・Pepper内の計算リソースで処理に時間がかかる場合は、外部システムで処理を行う。
・自動解放されないシステムリソースは終了時、エラー発生時に解放する。

NGになる例

・CPU利用率が著しく高く、バックグラウンドタスクの実行に支障をきたしている。
・アプリ終了後も関連プログラムが動作し続けている。
・メモリリークが発生している。

002-001-009　アプリ中断操作後のアプリ起動（複数回）

Conditions(対象)	-（全てのロボアプリ）
Expected(期待する動作)	アプリ中断操作(4パターン)を組み合わせ、アプリ中断操作後に再度アプリが起動できることを〜10回繰り返す - 急激な操作、胸ボタン2回押し、頭長タッチ、スリープ

テスト方法

・急激な操作、胸ボタン2回押し、頭長タッチ、スリープの4パターンの操作によりアプリを停止させ、全てのパターンで正常に起動できることを確認する。

実装例

・内部データの不整合によるエラーハンドリングを適切におこなう。

NGになる例

・起動中に意図しない形でデータ（ロックファイル等）が残され起動できない。
・エラー等でアプリが強制終了してしまった場合、以後起動できない。
・不完全な前回データが元でアプリケーションが動作中にスタックしてしまう。

002-001-011　カメラ機能利用時の負荷及びエラー対策

Conditions(対象)	カメラ利用の機能(バーコード/QRコードリーダー)を実装するアプリが対象
Expected(期待する動作)	・カメラ機能を利用する場合、負荷耐久試験やエラーハンドリング対策を入念に実施すること - 繰り返し読み込ませてもメモリリークが発生しないことを確認する - エラーから復帰した後も機能が正常に動作することを確認する - 読み込み不能時のエラーハンドリングが行われていることを確認する

テスト方法

・カメラを利用する場合、以下を実施する。

- アプリによるメモリリーク有無の確認
- エラーハンドリングの確認
- 十分な負荷試験

実装例

・カメラ利用時にはタイムアウトを設けて、連続してカメラに負荷がかからないようにする。

NGになる例

・QRコードの認識失敗などにより、カメラを利用し続けアプリがフリーズしてしまう。
・QRコードの認識失敗により再起動が必要になってしまう。
・他のアプリからカメラが利用できなくなってしまう。

003-005-002　アプリ操作中にNW切断

Conditions(対象)	NWへの接続があるアプリが対象
Expected(期待する動作)	・アプリ操作中、NWへ接続する直前にNWを切断しても、フリーズなどの異常動作が起こらないこと ・NWへの接続がなくアプリを終了する場合、終了する旨のメッセージをユーザへ通知した上でアプリを正常終了すること

テスト方法

・NW通信を行なっているタイミングでネットワークの切断を行い、アプリがスタック・あるいは利用不可能にならないことを確認する。

※NW接続を必要とするすべての画面で確認する
※NWを使用するAPIなどを実行する直前、直後も確認する

実装例

・レスポンスにタイムアウトを設定し、NWが切断された場合でもアプリの進行が止まらないようにする。
・コンテンツのダウンロードを行うアプリでは、ダウンロードしたコンテンツの完全性を検証する適切なエラーハンドリングを行う。

NGになる例

・NW経由で不完全にダウンロードされたコンテンツ、あるいは破損したコンテンツが原因でアプリの利用が不可能になってしまう。
・通信の遮断により、アプリがタイムアウトせず、スタックしてしまう。

003-005-003　タブレットと本体が別APに接続

Conditions(対象)	NWへの接続があるアプリが対象
Expected(期待する動作)	タブレットと本体が別APに接続している場合でもアプリが動作すること

テスト方法

・タブレットと本体を別のAP（アクセスポイント）に接続させ動作することを確認する。

別のAPと接続させる手順は次ページ参照

（別のAPに接続されることによるエラーハンドリングが適切に組み込まれていることを確認する）

実装例

・異なるAPに接続した状態でも動作するようにエラーハンドリングを適切に行う。

NGになる例

・以下の状況下でアプリが正常に動作しない。（スタックしてしまう）

003-005-003　タブレットと本体が別APに接続（手順）

前提

2つのネットワーク（それぞれに異なるSSIDとVLAN）を構成します。

手順

1. 設定アプリを起動しWi-Fiネットワークに接続します。
2. SSHでPepperに接続します。
3. 以下のコマンドを入力し、設定アプリ上で設定したネットワークとは違うネットワーク情報をタブレットに設定します。

書式> `qicli call ALTabletService.conigureWifi<security_type> <SSID> <PASSWORD>`
例> `qicli call ALTabletService.configureWifi WPA2 bigwave secret`

4. 以下のコマンドを入力し、設定したネットワークに接続します。

書式> `qicli call ALTabletService.connectWifi<SSID>`
例> `qicli call ALTabletService.connectWifi bigwave`

5. 以下のコマンドを入力し、設定したネットワークに接続できたか確認します。

```
> qicli call ALTabletService.getWifiStatus
"CONNECTED" ←接続が確認できます。
```

機体：SSID_A

タブレット：SSID_B

※SSID_Bの接続が変化する

004-002-002　ASRの動作

Conditions(対象)	ユーザーの音声を聞き取る箇所があるアプリが対象
Expected(期待する動作)	ASRの動作に不安定さが無いこと ASRの起動停止が頻繁に繰り返されていないこと

テスト方法

・音声認識モードと発話モードが適切に切り替わり、且つ不安定な動作をしないことを確認する。
・音声認識と発話が同時に行われていないことを確認する。

実装例

・音声認識中と音声出力中を**完全に分離し**適切に切り替える。

ポイント:

ASRとはAutomatic Speech Recognitionの略で
自動音声認識のこと。
音声認識起動中に、
Pepperの発話や音声再生が行われると
それ自体を認識しようとするため、
正常に音声認識できなくなる場合がある。

NGになる例

・音声認識中に発話が行われている。
・発話中に音声認識が行われている。
・音声認識と発話が頻繁に切り替わる。

004-005-001　危険な動作

Conditions(対象)	-（全てのロボアプリ）
Expected(期待する動作)	ユーザーに危害(物理的/心理的)を加えるような動作がないこと

テスト方法

・近づいてセーフティが動作するか確認する。
・動作の中に恐怖心を与えていないか確認する。
・複数人で動作が危険ではないか確認する。
（不快→日本人の感覚で）

確認箇所：
スピード、角度、状態維持時間、動作の組み合わせ、動作空間、パーツ単位（顔、手、腕、二の腕、肩、腰）

実装例

・セイフティーを無効化するAPIをコールしない。
・それぞれのポーズに遷移する前にマージン（時間）をもうける。
・個々のモーションの中に著しいモータ音が発生しないことを確認し、モーションを作成する。

NGになる例

・前に立っているユーザにぶつかる動作（勢いがあっても当たらない動作）が含まれている。
・モータ音がなる動作が含まれている。
・暴力的な動作が含まれている。

004-005-005　バランス

Conditions(対象)	-（全てのロボアプリ）
Expected(期待する動作)	3つのホイールが床に接地していること PushRecovery機能が動作していないこと

テスト方法

・アプリの動作によってロボット自体の姿勢バランスを崩すことがないことを確認する。
・ホイールが常に接地している事を確認する。

実装例

・バランスを崩す無理なモーションを行わない。
・セーフティを切らない

NGになる例

・Pepperには転倒を回避するためのPushRecovery機能が動作してしまう。

005-002-006　タブレットの日時取得

Conditions(対象)	日時を取得するアプリが対象
Expected(期待する動作)	タブレット基準の日時取得を行っていないこと

テスト方法

・日時取得する際タブレット側ではなく、機体側で処理を行なっていることを確認する。

実装例

・ディスプレイで日時情報を利用する際はALMemory経由で機体側より日時取得する。

時間の取得

NGになる例

・タブレット側の日時情報を利用している。

005-004-002　Preferenceの使用有無

Conditions(対象)	事前にPreferenceを使用する承諾を得ているアプリが対象 ※Pepper for Bizのロボアプリ公開申請フォームから申し込むアプリは対象外
Expected(期待する動作)	・承諾を得た範囲内でPreferenceDBの書き込みやPreferenceを使用していること

テスト方法

・ビヘイビア及び関連するライブラリファイル、そのほかの構成ファイルがALPreferenceManagerを呼び出の有無を確認する。

※PreferenceはALPreferenceManagerをさします。

実装例

・ALPreferenceManagerの利用の許諾が得られない、あるいは対象外であった場合は、ALPreferenceManager以外の方法（アプリ内あるいはクラウド等）でデータを保存する。
・ALPreferenceManager利用の許諾の範囲内になっているか確認する。

NGになる例

・ALPreferenceManagerの利用の対象外にかかわらず、同APIを利用している。

005-004-004　不適切な情報の保存有無

Conditions(対象)	事前にPreferenceを使用する承諾を得ているアプリが対象 ※Pepper for Bizのロボアプリ公開申請フォームから申し込むアプリは対象外
Expected(期待する動作)	・PreferenceDBに以下のような情報が含まれていないこと － ID/PASSなどのセキュリティ情報 － 個人情報などの保護すべき情報 － その他、公開してはいけない情報（非公開のURL、SSIDなど）

テスト方法

・ALPreferenceManagerを利用し秘匿すべき情報を保存してないことを確認する。

※PreferenceはALPreferenceManagerをさします。

実装例

・秘匿すべき情報は適切に暗号化処理を行い、クラウドなどのセキュリティが担保できるストレージに保存する。

NGになる例

・ALPreferenceManagerを利用しパスワードを保存している。

006-002-009　対象NAOqiのMaxVersion

Conditions(対象)	-（全てのロボアプリ）
Expected(期待する動作)	最大で"2"を指定していること

テスト方法

・manifestの記述内容を確認する。

実装例

・manifestを以下のように記述する。

```
<naoqiRequirement maxVersion="2"
minVersion="2.5"/>
```

minVersion より maxVersion が低く見えますが、“2” を指定することで、2.x 全てのという意味になります

NGになる例

・maxVersionの記述がない。

```
<naoqiRequirement minVersion="2.5"/>
```

・記述が不正。

```
<naoqiRequirement maxVersion="2.4"
minVersion="2.5"/>
```

007-001-002　エラーログ出力有無

Conditions(対象)	-（全てのロボアプリ）
Expected(期待する動作)	エラーログが出力されないこと

テスト方法

・ログを出力を確認する。

実装例

・正常動作の状態でエラーが出力しない。

NGになる例

・実行中にエラーログが出力される（アプリのクラッシュの有無にかかわらず審査NG）

007-001-003　アプリ独自のログ出力有無

Conditions(対象)	-（全てのロボアプリ）
Expected(期待する動作)	不具合解析に必要な以下のログ以外の独自ログが出力されていないこと ・エラーを検出するためのログ(例外処理時) ・外部連携を行う際にレスポンスを確認するログ ・Service等の登録を確認するログ 上記ログ出力させる際に、大量のログが出力されない作りになっていること

テスト方法

・self.loggerを使ったログ出力を行なってないことを確認する。

実装例

・コーディングしたself.loggerを削除する（あるいは実装しない）。

NGになる例

・以下のようなステートメント独自に作成されたボックスに残っている。

```
self.logger.info('log strings')
```

007-001-004　不適切なログ出力内容の有無

Conditions(対象)	-（全てのロボアプリ）
Expected(期待する動作)	・出力するログに以下のような情報が含まれていないこと - ID/PASSなどのセキュリティ情報 - 個人情報などの保護すべき情報 - その他、公開してはいけない情報(非公開のURL、SSIDなど)

テスト方法

・ログ出力に秘匿すべき情報が含まれてないことを確認する。

実装例

・ログ出力中に秘匿すべき情報が含まれていた場合、該当部分を削除する。

NGになる例

・ログ出力にWiFiに接続するSSID及び事前共有キーが含まれている。

007-001-005　データ保存場所

Conditions(対象)	アプリ内でデータ保存するアプリが対象
Expected(期待する動作)	アプリ内でデータ保存する場合はアプリフォルダ内(UUID配下)に保存していること ※ソース上でもデータ格納先を確認すること

テスト方法

・アプリフォルダにデータが正しく保存されていることを確認する。

実装例

・アプリフォルダを取得し、データを保存する。

参考:

```
# アプリフォルダを取得する
appFolder =
self.behaviorAbsolutePath().replace(self.beha
viorRelativePath(), "")
```

NGになる例

・アプリフォルダ外の書き込み可能な任意のフォルダにデータを書き込んでいる。

/home/nao 等

007-001-025　タイムアウト時の戻り値

Conditions(対象)	お仕事かんたん生成2.0から連携して起動するアプリが対象
Expected(期待する動作)	ALMemory “PFB/P4BA/OutputParameter” にお仕事の先頭に戻る以外の用途で,9999 を設定しないこと。 ※タイムアウトなどの場面で使用してください ※“PFB/P4BA/OutputParameter” に値(1～99)を書き込むと2.0復帰時に参照されます。

テスト方法

・戻り値を確認する

実装例

・戻り値を”9999”に設定している場合、お仕事の先頭に戻る以外の意図で使用していないこと

参考:

例:OutputParameterに1を書き込む

```
self.memory = ALProxy("ALMemory")
self.memory.insertData("PFB/P4BA/OutputParameter",1)
```

NGになる例

・値が不正。

```
self.memory.insertData("PFB/P4BA/OutputParameter",9999)
```

End of File