Speak Unity SDK User's Guide

Ver. v1.14.4

Rev. 9a986570864d952f1dca0c27e7b9e23edc8412a1

はじめに

本章で記載の内容は，NTT DOCOMO INC.（以下「当社」といいます）が提供する本マニュアルを利用されるすべての方（以下「利用者」といいます）に適用されます．利用者は，本章をよくお読みいただき，ご了承の上，本マニュアルをご利用願います．当社は，本マニュアルのご利用をもって，本章記載内容をご了承いただいたものと見なすものとします．なお，下記内容につきましては予告なく変更させていただく場合がありますので，あらかじめご了承ください．

免責事項

1. 本マニュアルの内容は，情報の提供のみを目的とするものです．本マニュアルあるいは本書に記載された内容や製品に関して情報の全部または一部を予告なく変更する場合があります．また，本マニュアルの提供を休止または停止する場合があります．
2. 当社は，本マニュアルあるいは本書に記載された内容や製品に関して，明言または保証するものではありません
3. 当社は，本マニュアルあるいは本書に記載された製品の使用による直接的間接的あるいは事故による，結果的あるいは特別な行為による一切の損失，損害を保証しません
4. 当社は，お客様が本マニュアルからリンクが張られている第三者のウェブサイト，または本マニュアルを張っている第三者のウェブサイトから取得された各種情報のご利用によって生じたいかなる損害についても責任を負いません

著作権と商標について

1. 本マニュアル上に掲載されている著作物（文書・写真・イラスト・動画・音声・ソフトウェア等）の著作権は，当社または第三者が保有しており，著作権法その他の法律ならびに条約により保護されています
2. 私的使用目的の複製，引用など著作権法上認められている範囲を除き，著作権者の許諾なしに，これらの著作物を複製・翻案・公衆送信等することはできません

準拠法および管轄裁判所

1. 本マニュアルは当社の管理下にあります．本マニュアルは，法律の異なる全世界の国々からアクセスすることが可能ですが，利用者および当社の両者は，本マニュアルの利用に関して日本国の法律および東京都の条例に拘束されることに同意するものとします
2. 当社は本マニュアル上で，利用者環境において本マニュアルのコンテンツが適切であるかなどの記述や表示は一切行いません
3. 本マニュアルの利用は，利用者の自由意志によるものとし，本マニュアルの利用に関しての責任は利用者にあるものといたします
4. 本マニュアルの利用に関わる全ての紛争については，別段の定めのない限り，東京地方裁判所を第一審の専属管轄裁判所とするものとします

本マニュアルへのリンクについて

本マニュアルへのリンクは，営利，非営利を問わず原則自由とし，事前の当社へのご連絡は不要です．ただし，以下のいずれかに該当するか，またはそのおそれがあるリンクの設定はご遠慮ください．

1. 当社および当社の商品・サービス等を誹謗中傷したり，当社の信用を毀損する内容を含むウェブサイトからのリンク

2. 違法である情報を含むウェブサイトからのリンク
3. 公序良俗および社会倫理に反する内容を含むウェブサイトからのリンク
4. フレームリンク，イメージリンクなど，本マニュアルの情報であることが不明確となるリンク（ブラウザ画面全体が本マニュアルに変わるか，別画面が開いて本マニュアルが表示される形式でリンクを設定してください）
5. 当社と何らかの提携または協力関係にあるとの誤解を与える，または当社がリンク元のウェブサイトを認知または支持しているとの誤認を生じさせるリンク

上記のいずれにも該当しない場合でも，リンクの設定方法の変更またはリンクの削除をお願いする場合があります．また，本マニュアルの URL は，予告なく変更または削除する場合があります．

謝辞

- This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)
- This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)

ソフトウェア開発キットの利用に関する規約

ソフトウェア開発キットの利用に関する規約（以下「本規約」という）は，株式会社ＮＴＴドコモ（以下「当社」という）が提供するソフトウェア開発キット（Speak SDK（オブジェクトコード））の使用条件等を定めるものである．貴社（以下「開発者」という）は本規約に同意しない場合，本件ソフトウェア開発キットを利用することはできないものとする．

第１条（定義）

本規約において使用する用語の定義は，次の各号に定めるとおりとする．

（1）「Speak」とは，自然言語処理技術によって自然な対話を通したサービス提供を可能とする技術をいう．
（2）「本件ソフトウェア開発キット」とは，外部サーバに対してあらかじめ取得した認証トークンや，エンドユーザから入力された音声・テキストのいずれかを送信し，外部サーバから送信された音声やテキストを受信する機能を持つ，Speak SDK（オブジェクトコード）をいう．
（3）「開発者」とは，本件ソフトウェア開発キットを使用又は複製して本件アプリを開発する貴社をいう．
（4）「本件アプリ」とは，本件ソフトウェア開発キットを使用又は複製し開発する，Speak 技術を利用したサービスのためにスマートフォン・タブレットその他機器上で動作するアプリケーションプログラムをいう．
（5）「サービス提供者」とは，本件アプリを使用して Speak 技術を利用したサービスを提供する事業者をいう．
（6）「エンドユーザ」とは，各サービス提供者が提供するサービスを利用する者をいう．

第２条（本規約への同意及び契約の成立）

開発者は本件ソフトウェア開発キットを自己のパーソナルコンピュータ等の機器上で実行した時点で本規約に同意したものとし，当社との間で本規約に基づく契約が成立し，その効力が生じるものとする．

第３条（利用許諾）

当社は開発者に対し，開発者が本規約の各条項を遵守することを条件に，開発者又はサービス提供者が日本国内で Speak 技術を利用したサービスを提供するにあたり必要となる本件アプリを開発し，本件アプリを開発者又はサービス提供者がエンドユーザに配布する目的に限り，本規約に従って次の各行為を行うことのできる，非独占的且つ譲渡不能な権利を許諾する．

（1）本件ソフトウェア開発キットを使用及び複製し，本件ソフトウェア開発キットを組み込んだ本件アプリの開発
（2）前号に基づき開発した本件アプリのサービス提供者への提供，及び開発者がサービス提供者に対して本件アプリと一体として本件ソフトウェア開発キットを使用する権利の再許諾
（3）開発者又はサービス提供者による本件アプリのエンドユーザへの配布，及び開発者がサービス提供者に対し，本件アプリと一体として本件ソフトウェア開発キットを複製しエンドユーザに配布できる権利の再許諾
（4）エンドユーザによる本件アプリの使用，及び開発者がサービス提供者をして又は開発者が直接エンドユーザに本件アプリと一体として本件ソフトウェア開発キットを使用できる権利の再許諾

２．前項に基づき開発者がサービス提供者及びエンドユーザに再許諾する権利は，以下の制限を受けるものとする．

（1）サービス提供者は，日本国内で Speak 技術を利用したサービスを提供する範囲において本件アプリを使用，複製及び配布する限りにおいてのみ，本件アプリに組み込まれた本件ソフトウェア開発キットを使用，複製及び配布できるものとする．
（2）エンドユーザは，サービス提供者が提供するサービスを利用する範囲において本件アプリを使用する限りにおいてのみ，本件アプリに組み込まれた本件ソフトウェア開発キットを使用できるものとする．
（3）開発者，サービス提供者及びエンドユーザは，本ソフトウェア開発キットに第三者の著作権が含まれていることを認識し，本ソフトウェア開発キットの全ての利用及び複製にあたり別紙 LICENSE_ThirdParty.txt に定めるものを含む本件ソフトウェア開発キットの著作権表示及び許諾条件を遵守するものとする．
（4）開発者は，本規約において自己が負うのと同等の義務をサービス提供者に課し，サービス提供者をして又は開発者が自らエンドユーザに課させるものとし，当該義務違反について当社に対し，一切の責を負うものとする．

第４条（本件ソフトウェア開発キットの変更）

当社は，当社の判断により，開発者への事前の通知なく本件ソフトウェア開発キットの全部又は一部を変更，追加，廃止，提供中断又は中止（以下「変更等」という）することができるものとする．
２．本件ソフトウェア開発キットの変更等により開発者又は第三者に生じた損害について，当社は一切の責任を負わない．

第５条（対価）

当社は開発者に対し本件ソフトウェア開発キットを無償で提供する．

第６条（非保証及び免責）

当社は開発者に対し，本件ソフトウェア開発キットを現状有姿で提供し，本件ソフトウェア開発キットの技術的正確性，実現性，市場性，特定目的適合性，第三者の権利侵害の有無等につき，いかなる明示的，黙示的な保証も行わない．
２．本規約に関連して発生した損害について当社が負う一切の責任は，現実且つ通常の損害に限られ，且つ，その上限は当社が開発者より現実に受領した対価の額とし，当社は開発者，サービス提供者，エンドユーザ，その他の第三者に対し，いかなる場合においても，特別損害，間接的損害，付随的損害及び拡大損害（逸失利益，機会損失等を含む）について一切の責任を負わない．

第７条（エンドユーザからの同意の取得）

開発者は，自ら又はサービス提供者が本件アプリをエンドユーザに対し使用させる場合，以下に定める事項について，エンドユーザの個別且つ明確な同意を事前に自ら取得し又はサービス提供者に取得させるものとし，エンドユーザのプライバシーを保護するために必要な措置を講じるものとする．

●開発者又はサービス提供者が本件アプリを通じて，本件アプリ毎に個別に払い出される認証トークンの他，エンドユーザが発話した音声・テキストのいずれかの情報を取得し，開発者，サービス提供者又は第三者のサーバ等に送信すること，及び当該開発者，サービス提供者又は第三者が当該情報を分析すること．
●前記のとおりサーバに送信されたデータは，Speak 技術を利用したサービスの安定的な稼働及び Speak 技術を利用したサービス，開発者及びサービス提供者のサービス向上のために利用すること．
●エンドユーザが発話した音声・テキストが開発者，サービス提供者又は第三者のサーバ等に送信され，当該情報が分析されることに鑑み，エンドユーザが自己及び第三者の個人情報を発話・入力しないこと．

第８条（禁止行為等）

開発者は，本件ソフトウェア開発キットの利用について，以下の各号のいずれかに該当する又は該当する可能性があると当社が判断する行為をしてはならない．

①本件ソフトウェア開発キットの全部又は一部をリバース・エンジニアリング，逆アセンブル，逆コンパイルその他の解析をする行為
②本件ソフトウェア開発キットの利用により，第三者を差別もしくは誹謗中傷・侮辱し，名誉を毀損し，又はプライバシーを侵害する行為
③本件ソフトウェア開発キットの利用により第三者の知的財産権（著作権，特許権，実用新案権，意匠権，商標権，肖像権を含む）その他の権利を侵害する行為
④当社又は第三者のネットワークその他の設備に過度な負担を与え，又は不正な動作をさせる行為
⑤その他本規約に違反する行為
⑥前各号の行為を推奨，助長又は幇助等する行為
⑦前各号の他，当社が不適当と認める行為
２．当社は本条を含め本規約に違反する本件ソフトウェア開発キットの利用を認知した場合は，事前の通知なく本件ソフトウェア開発キットの使用の差し止めを行い，本規約に基づく契約を解除することができるものとする．

第９条（第三者との紛争）

本件ソフトウェア開発キットの利用に関して，開発者と第三者（サービス提供者，エンドユーザを含むがこれに限らない）との間で紛争等が発生した場合，当社は当該紛争等の解決義務を負わないものとし，開発者は開発者自身の費用と責任で当該紛争等を解決するものとする．また，当社が任意に当該紛争等の解決努力をした場合でも，解決義務および継続的な解決努力義務を負うものではない．
２．前項の紛争および開発者が本規約に違反したことにより，当社が損害（対応に要した弁護士費用等を含む）を被った場合，開発者は当社に対して当該損害を賠償し，当社が支出した一切の費用（合理的な範囲の弁護士費用等を含む）を補償するものとする．

第１０条（知的財産権等）

本規約による本件ソフトウェア開発キットの利用許諾は，当社から開発者，サービス提供者又はエンドユーザへの本件ソフトウェア開発キットの著作権，その他何らの権利を移転するものではない．
２．開発者は，本規約に基づく契約の成立は，本規約に明示的に規定されているものを除き，当社が開発者，サービス提供者，エンドユーザ，その他の第三者に対して，当社の保有する特許権，実用新案権，意匠権，商標権，著作権，技術上又は営業上のノウハウ若しくはその他の権利，又はこれらを受ける権利に基づく実施権等の権利を許諾するものではないことを確認し，同意する．

第１１条（権利の譲渡等）

開発者は，本規約に基づき当社に対して有する権利又は当社に対して負う義務の全部又は一部を第三者に譲渡し，承継させ，又は担保に供することはできない．

第１２条（本規約の変更）

当社は，当社の都合により開発者への事前の通知又は承諾を得ることなく，本規約を変更することができる．この場合，変更後の本規約が適用されるものとし，変更後の本規約は当社が別途定める方法にて開発者に通知されるものとする．

第１３条（合意管轄）

当社と開発者との間で本規約に関連して訴訟の必要が生じた場合は，東京地方裁判所を第一審の専属的合意管轄裁判所とする．

第１４条（準拠法）

本規約に基づく契約の成立，効力，解釈および履行については，日本国法に準拠するものとする．

附則

本規約は，2019 年 4 月 18 日から適用します．

以上

2020/4/22

目次

第 1 章

動作環境

SpeakSDK での Unity 対応バージョンを下記に示す.

動作プラットフォーム	対応バージョン
Android	2019.3.2f1
iOS	
LuminOS	
Windows	
macOS	

1.1 Android

SpeakSDK では以下の動作環境を対象としている. また端末スピーカー及び端末マイクを用いた動作のみを保証する.

Android	API level	アーキテクチャ
5.0~10.0	21~29	armeabi-v7a, arm64-v8a

ハードウェア動作推奨スペックを下記に示す.

CPU クロック周波数	CPU コア数	メモリ
1.2GHz 以上	2 以上	50MB 以上

試験実施により動作確認済みの環境を下記に示す.

端末名	Android	API level	アーキテクチャ
Oculus Quest	7.1.1	25	arm64-v8a

各開発環境の対応バージョンを下記に示す.

開発環境	対応バージョン
Android Studio	3.4.2
NDK	r19
SDK	r21

1.2 iOS

SpeakSDK では以下の動作環境を対象としている．また端末スピーカー及び端末マイクを用いた動作のみを保証する．

iOS	アーキテクチャ
10.0~13.0	32bit, 64bit

ハードウェア動作推奨スペックを下記に示す．

CPU クロック周波数	CPU コア数	メモリ
1.2GHz 以上	2 以上	50MB 以上

試験実施により動作確認済みの環境を下記に示す．

端末名	iOS	アーキテクチャ
iPhone11	13.2.3	64bit

開発環境の対応バージョンを下記に示す． ただし，iOS シミュレータ上での動作は未対応となっている．

開発環境	対応バージョン
Xcode	11.0

1.3 LuminOS

SpeakSDK では以下の動作環境を対象としている．また端末スピーカー及び端末マイクを用いた動作のみを保証する．

LuminOS	アーキテクチャ
0.98.10	arm64-v8a

試験実施により動作確認済みの環境を下記に示す．

端末名	LuminOS	アーキテクチャ
Magic Leap One	0.98.10	arm64-v8a

開発環境の対応バージョンを下記に示す．

開発環境	対応バージョン
LuminSDK	0.24.10

1.4 Windows

SpeakSDK では以下の動作環境を対象としている．また端末スピーカー及び端末マイクを用いた動作のみを保証する．

Windows	アーキテクチャ
7~10	32bit, 64bit

ハードウェア動作推奨スペックを下記に示す.

CPU クロック周波数	CPU コア数	メモリ
3.6GHz 以上	4 以上	64bit

試験実施により動作確認済みの環境を下記に示す.

端末名	Windows バージョン	アーキテクチャ
Lattitude 5580	Windows10 1903	64bit

1.5 macOS

SpeakSDK では以下の動作環境を対象としている. また端末スピーカー及び端末マイクを用いた動作のみを保証する.

macOS	アーキテクチャ
10.9〜10.15	364bit

ハードウェア動作推奨スペックを下記に示す.

CPU クロック周波数	CPU コア数	メモリ
1.4GHz 以上	4 以上	64bit

試験実施により動作確認済みの環境を下記に示す.

端末名	macOS バージョン	アーキテクチャ
MacBook Pro	10.15.2	64bit

第 2 章

クイックスタートガイド

以下のサンプルコードによって音声対話を行うことができる.

```
public class SpeakSDKManager : MonoBehaviour
{
    void Start()
    {
        Speak.Instance().SetURL("wss://hostname.domain:443/path");
        Speak.Instance().SetDeviceToken("xxxxxxxx");
        Speak.Instance().Start(null, null);
    }

    void Update()
    {
        try
        {
            Speak.Instance().Poll();
        }
        catch
        {
            //exception 発生時の処理
        }
    }

    void OnApplicationQuit()
    {
        Speak.Instance().Stop(OnStop);
        while (Speak.Instance().Poll(true)) { }
    }
}
```

2.1 iOS

マイク権限の設定のため下記を設定する.

Player Settings > Other Settings > Configuration > Microphone Usage Description にマイクを使用する目的を記述する.

アプリ起動時に上記で設定した説明文と共に使用権限を取得するダイアログが表示され，許諾後に音声対話が可能となる.

2.2　LuminOS

プロジェクトの設定を参照し，MagicLeap 向け Unity アプリの設定を行う．
Project Settings の Manifest Settings から下記権限を有効にする．権限の詳細に関しては Manifest Settings の設定項目を参照のこと．

- LowLatencyLightwear
- AudioCaptureMic
- Internet

アプリをインストール後，端末のトップ画面から Settings > Privacy > Privileges を選択し，当該アプリに対するマイク権限 (AudioCaptureMic) を有効にする．

権限設定後，アプリを起動すると音声対話が開始する．

第 3 章

Speak とは

Speak とは音声対話に必要な様々な機能をシンプルなインターフェースで開発者に提供し，音声対話アプリケーションを容易に実現することを目的として設計された SDK である．

3.1 システム全体構成

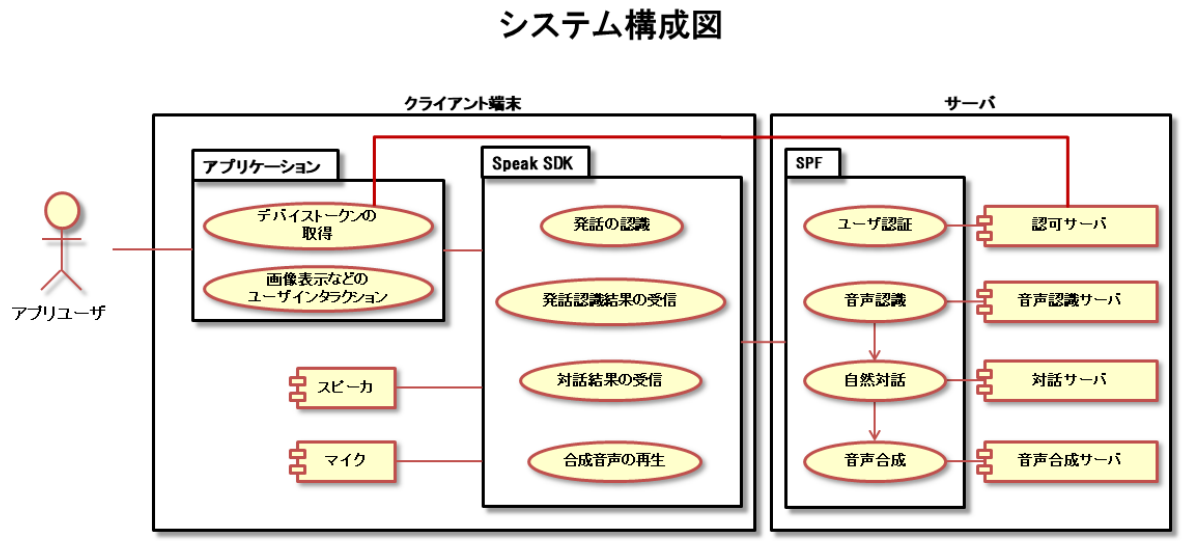

図 3.1　システム構成図

Speak はマイクやスピーカといったプラットフォーム固有の音声デバイス制御を内包している．そのため，アプリケーションの開発者はデバイスの詳細を意識せずに，簡易なインターフェースで音声ミュートや合成音再生といった機能を利用することができる．

Speak は起動すると自動的にサーバへ接続し，ユーザの発話音声データまたはテキスト入力データの送信が可能な状態となる．サーバのフロントエンドを SPF と呼ぶ． SPF は以下のサーバ群に接続されており，音声対話に関する一連の機能を実現している．

- 認可サーバ (UDS)：デバイストークン払い出し機能及びユーザ認証機能を有する．
- 音声認識サーバ (ASR)：ユーザ発話音声を認識し，文字列として変換する機能を有する．
- 対話サーバ (NLU)：ユーザ発話の内容，またはテキストデータからシステムの応答データを生成する機能を有する．
- 音声合成サーバ (TTS)：システムの応答データから，合成音を生成する機能を有する．

3.2 対話の流れ

サーバへ接続後の対話の基本的な流れは以下の通りである.

1. クライアント端末からサーバに音声データまたは自然文が送られる
2. 音声データの場合は，音声認識が実行されて，ユーザ発話文字列に変換される
3. 音声認識結果がクライアント端末に返却される [*1]
4. ユーザ発話文字列または自然文とメタ情報 (対話のための付加的なデータ) を入力として，対話が実行される
5. 対話の結果 (NLU_result) が生成される
6. クライアント端末に NLU_result が返却される [*2]
7. NLU_result にシステム発話文字列が含まれる場合は，システム発話文字列が音声データに変換される
8. クライアント端末に音声データが返却される
9. クライアント端末でシステム発話文字列の音声データが再生される

[*1] 返却される音声認識結果からユーザ発話文字列を取得する方法はテキスト情報の受信を参照のこと.
[*2] 返却される NLU_result の詳細及び NLU_result からシステム発話文字列を取得する方法は，メタ情報の受信のメタ情報の受信を参照のこと.

第 4 章

SDK のライフサイクル

SDK には以下の 4 つの状態が存在する． 各状態の詳細は後述の通りである．

- 停止状態
- 接続の開始中
- 接続状態
- 接続の停止中

4.1 停止状態

SDK のインスタンス生成直後はこの状態から開始する．
SDK に対するセットアップはこの状態の時に行う必要がある．(参照:セットアップに関するサンプルコード)
一度セットアップした設定は保持されるため, 停止のたびに再セットアップする必要はない．
接続の開始が実行されると「接続の開始中」に移行する．(参照:接続の開始に開始に関するサンプルコード)

4.2 接続の開始中

SDK 内部で初期化処理およびサーバへの接続処理が行われる状態である．サーバへの接続が完了すると「接続状態」に移行する．この時アプリに onStart イベントが通知される．初期化処理とサーバへの接続処理は並行して行われるため，onStart イベントが通知された時に初期化処理が完了しているとは限らない．初期化処理またはサーバへの接続処理中にエラーが発生すると「停止状態」に移行する．この時アプリに onFailed イベントが通知される．

4.3 接続状態

SDK を通して対話が可能な状態である． 接続開始直後から音声入力が ON の状態となり, ユーザの発話は音声データとして自動でサーバに送信される．
この状態の時に，SDK 内部またはサーバ側でエラーが発生すると「停止状態」に移行する．この時アプリに onFailed イベントが通知される．接続の終了が実行されると「接続の停止中」に移行する．(参照:接続の終了に関するサンプルコード)

4.4 接続の停止中

SDK 内部でサーバへの接続の停止処理が行われている状態である．接続の停止が完了すると「停止状態」に移行する．この時アプリに onStop イベントが通知される．接続の停止処理中にエラーが発生すると「停止状態」に移行する．この時アプリに onFailed イベントが通知される．

4.5 onStart/onStop イベントに関する注意点

「停止状態」以外の状態でエラーが発生した場合は，アプリに onFailed イベントが通知される．この際，onStart または onStop イベントがアプリに通知されるとは限らないため，onStart および onStop イベントが通知される事を前提とした処理の実装には注意

が必要となる.

4.6　状態遷移図

上記 4 つの状態を状態遷移図としてまとめたものが下記の図である.

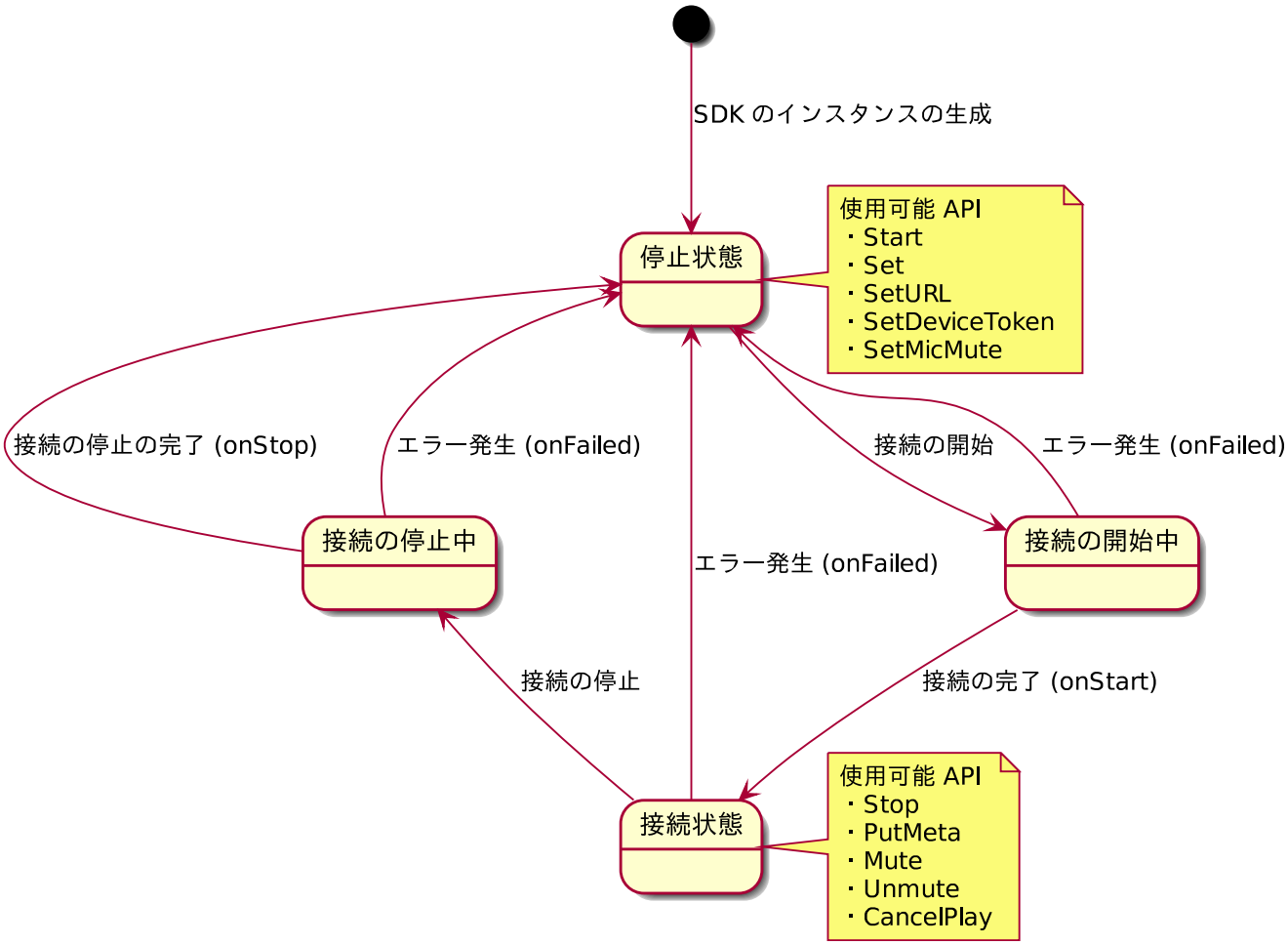

図 4.1　SDK のライフサイクル

第 5 章

プロジェクトの設定

5.1 unitypackage のインポート

1. Unity を起動し Unity プロジェクトを開く.
2. Assets > Import Package > Custom Package... にてインポートする unitypackage を選択する (※ 2).

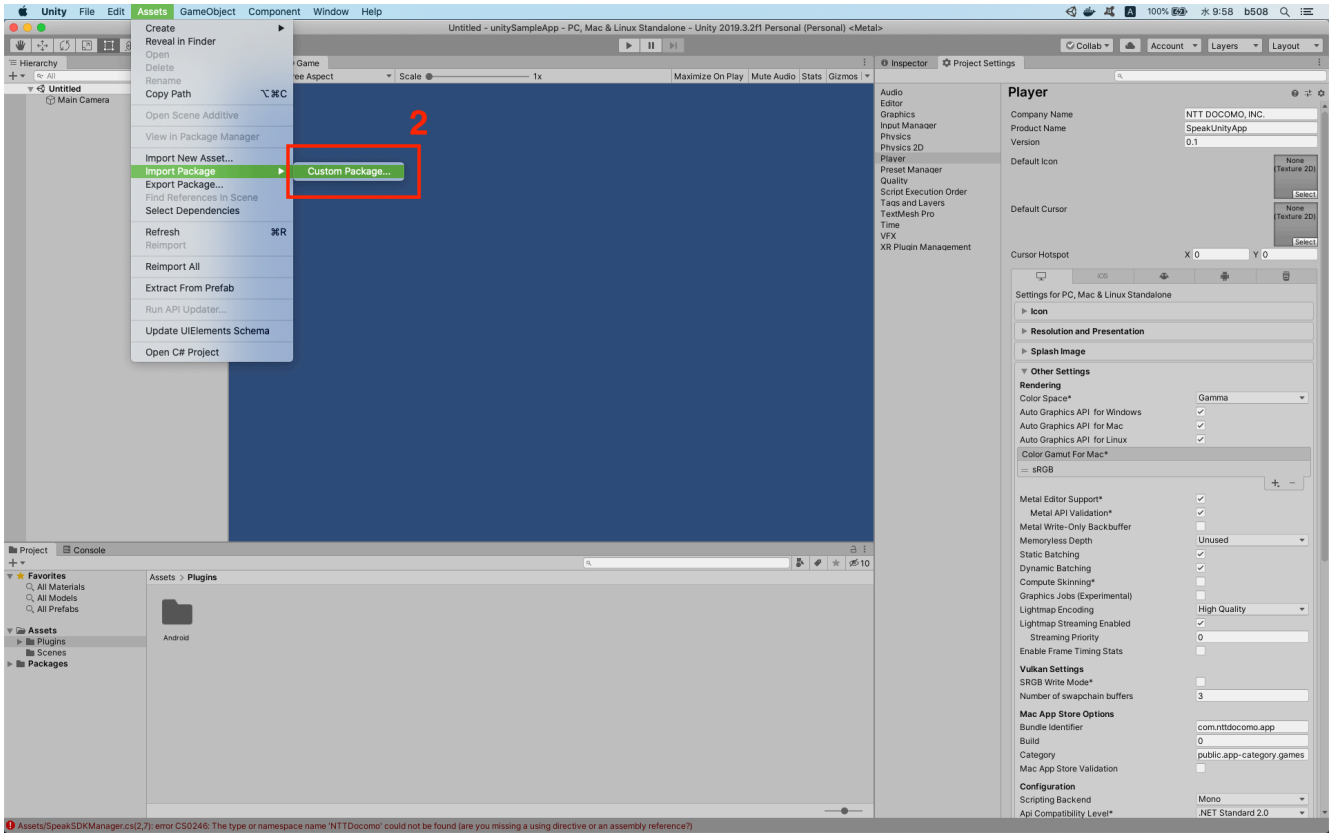

図 5.1 a

3. インポートするライブラリを選択し (※ 3-1)，import ボタンを押下する (※ 3-2).

4. Assets にライブラリがインポートされる (※ 4).

5.2 LuminOS

MagicLeap 向け Unity プロジェクトの準備

こちらを参照し，MagicLeap 向け Unity アプリのプロジェクトを作成する.

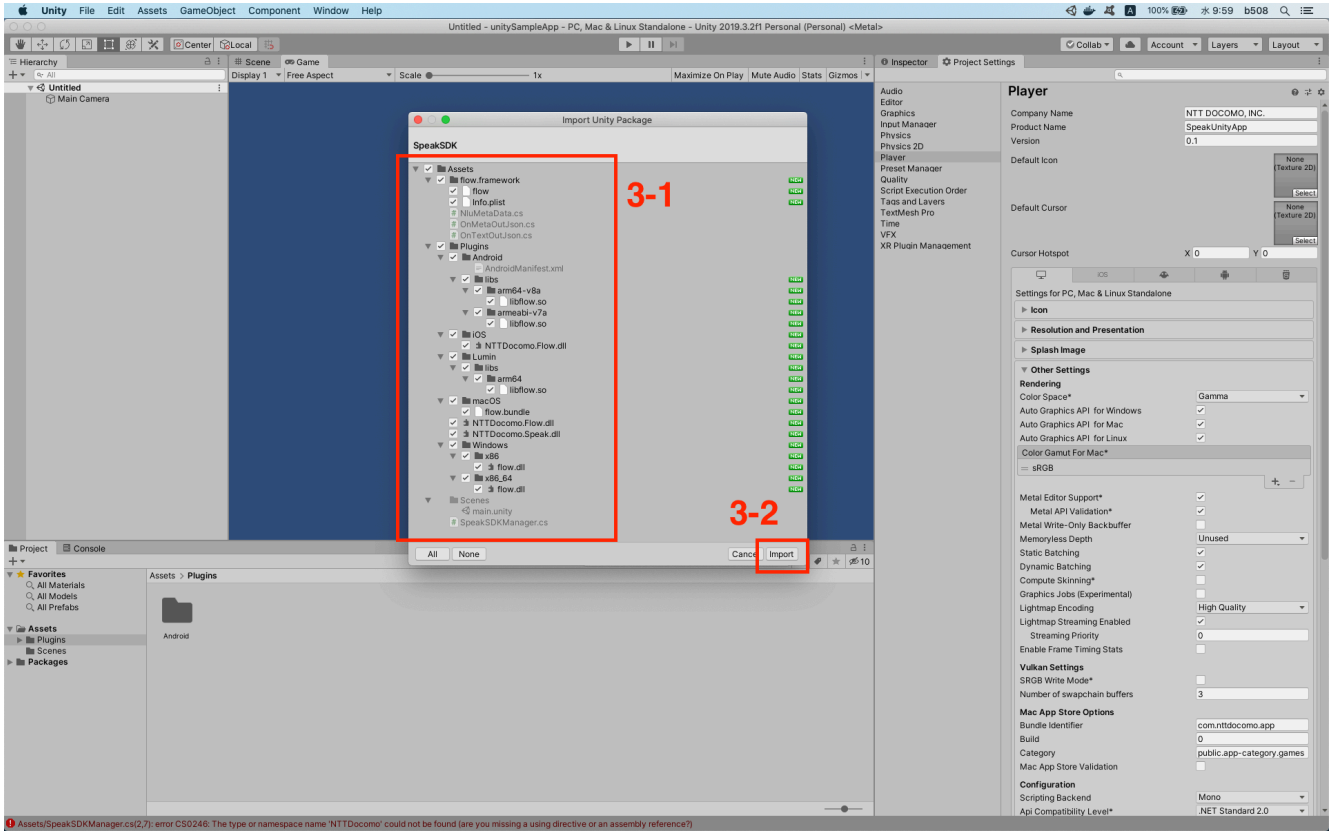

図 5.2　b

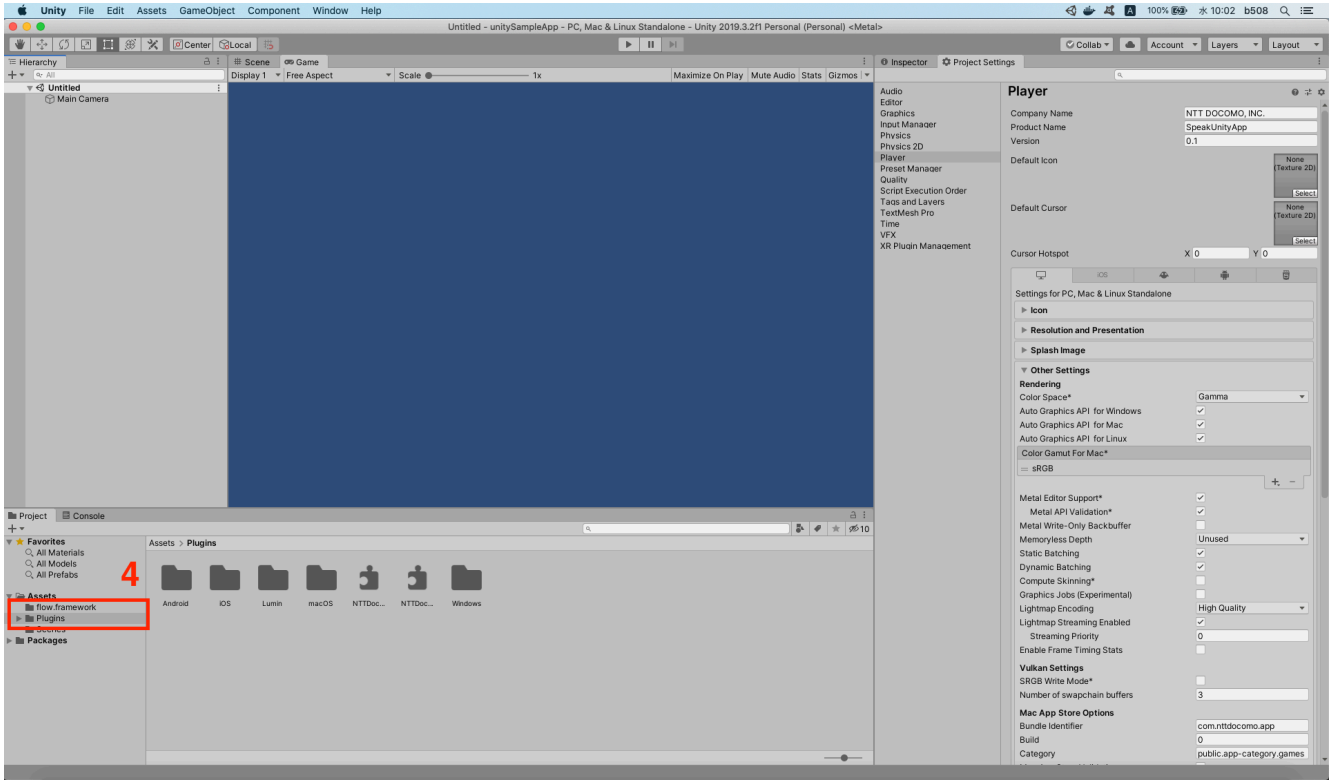

図 5.3　c

証明書の設定

こちらを参照し，証明書の設定を行う．

Manifest Settings の設定項目

本 SDK を利用するために最低限必要な user-permission を以下の表に示す．

パーミッション名	用途
LowLatencyLightwear	主要デバイス機能への接続
AudioCaptureMic	ユーザ発話の録音
Internet	インターネットへの接続

第 6 章

SDK のセットアップ

6.1 インスタンスの生成

本 SDK は複数の Speak クラスのインスタンスが生成されると動作しないため，シングルトンインスタンスを取得する．

```
using NTTDocomo.Speak;

Speak.Instance();

}
```

6.2 パラメータの設定

Speak#Set メソッドまたは Speak#SetXXX メソッドを用いて SDK に接続情報や認証情報などのパラメータを設定する．これらのパラメータの設定は接続の開始前に実行する必要がある．

設定必須のパラメータは以下の通り．

- 接続先の URL
- 認証用のデバイストークン

これらのパラメータは以下のサンプルコードのように設定する．

```
Speak.Instance().SetURL("wss://hostname.domain:443/path");
Speak.Instance().SetDeviceToken("XXXX");
```

6.3 接続の開始

Speak#Start メソッドでサーバに接続する． 第 1 引数には接続が完了した際に実行されるイベントハンドラを指定する． 第 2 引数にはエラーが発生した際に実行されるイベントハンドラを指定する．詳細なエラー内容については，トラブルシューティングを参照のこと．

以下のコードは，接続の開始のサンプルコードである．

```
public void OnStart()
{
    //  接続完了時に実行する処理を記述
}
```

```
public void OnFailed(int errorCode, string errorMessage)
{
    //  接続失敗時に実行する処理を記述
    //  第 1 引数（int errorCode）はエラーコード
    //  第 2 引数（String errorMessage）はエラーメッセージ
}

public void StartSpeakSDK()
{
    Speak.Instance().Start(OnStart, OnFailed);
}
```

Speak#Start メソッド実行時以外のタイミングで接続完了時のイベントハンドラを設定する場合には，Speak#OnStart を用いて設定を行う．

```
Speak.Instance().OnStart += OnStart;
```

Speak#Start メソッド実行時以外のタイミングでエラー発生時のイベントハンドラを設定する場合には，Speak#OnFailed を用いて設定を行う．

```
Speak.Instance().OnFailed += OnFailed;
```

6.4　イベントのハンドリング

SPF との接続中は，明示的に polling を実施する必要がある．接続開始後，Speak#Poll メソッドを呼び出すことで明示的に polling を行う．この際，呼び出しを MonoBehaviour#Update から行うことで，SDK からのイベント通知はすべて Unity メインスレッド上から通知される．(参照:SDK からのイベント通知)

```
public class SpeakSDKManager : MonoBehaviour
{
    void Update()
    {
        try
        {
            Speak.Instance().Poll();
        }
        catch
        {
            //exception 発生時の処理
        }
    }
}
```

6.5　接続の停止

Speak#Stop メソッドでサーバとの接続の停止を行う．引数には接続の停止が完了した際に実行されるイベントハンドラを指定する．Speak#Start メソッドで再接続が可能である．設定値の変更が不要な場合は，再セットアップも不要である．以下のコードは，接続の停止を行うサンプルコードである．

```
public void OnStop()
{
    //  接続停止完了時に実行する処理を記述
}

public void StopSpeakSDK()
{
    Speak.Instance().Stop(OnStop);
}
```

Speak#Stop メソッド実行時以外のタイミングで接続停止時のイベントハンドラを設定する場合には，Speak#OnStop を用いて設定を行う．

```
Speak.Instance().OnStop += OnStop;
```

第 7 章

SDK からのイベント通知

7.1 イベントハンドラの設定

音声再生開始と終了

音声再生開始時に何らかの処理を実行する場合は，Speak#SetOnPlayStart メソッドでイベントハンドラを設定する． イベントハンドラの引数 (string text) は常に空文字である．

```
public void OnPlayStart(string text)
{
    //  音声再生開始時に実行する処理を実装
}

void InitializeSDK()
{
    Speak.Instance().SetOnPlayStart(OnPlayStart);
}
```

また Speak#OnPlayStart を用いてイベントハンドラを設定することもできる．

```
Speak.Instance().OnPlayStart += OnPlayStart;
```

音声再生終了時に何らかの処理を実行する場合は，Speak#SetOnPlayEnd メソッドでイベントハンドラを設定する． イベントハンドラの引数 (string text) は常に空文字である．

```
public void OnPlayEnd(string text)
{
    //  音声再生終了時に実行する処理を実装
}
void InitializeSDK()
{
    Speak.Instance().SetOnPlayEnd(OnPlayEnd);
}
```

また Speak#OnPlayEnd を用いてイベントハンドラを設定することもできる．

```
Speak.Instance().OnPlayEnd += OnPlayEnd;
```

テキスト情報受信

音声認識結果受信時に何らかの処理を実行する場合は，Speak#SetOnTextOut メソッドでイベントハンドラを設定する．イベントハンドラの引数 (string text) には受信した音声認識結果が渡される．音声認識結果のフォーマットはこちらを参照のこと．

```
public void OnTextOut(string text)
{
    //  音声認識結果受信時に実行する処理を実装
}

void InitializeSDK()
{
    Speak.Instance().SetOnTextOut(OnTextOut);
}
```

また Speak#OnTextOut を用いてイベントハンドラを設定することもできる．

```
Speak.Instance().OnTextOut += OnTextOut;
```

メタ情報受信

メタ情報受信時に何らかの処理を実行する場合は，Speak#SetOnMetaOut メソッドでイベントハンドラを設定する．イベントハンドラの引数 (string text) には受信したメタ情報が渡される．(参照:メタ情報の受信)

```
public void OnMetaOut(string text)
{
    //  メタ情報受信時に実行する処理を実装
}
void InitializeSDK()
{
    Speak.Instance().SetOnMetaOut(OnMetaOut);
}
```

また Speak#OnMetaOut を用いてイベントハンドラを設定することもできる．

```
Speak.Instance().OnMetaOut += OnMetaOut;
```

7.2　テキスト情報の受信

Speak#SetOnTextOut で設定されたイベントハンドラにテキスト情報が返却される．現在，返却されるテキスト情報は音声認識結果の 1 種類のみである．

音声認識結果

テキスト情報として返却される音声認識結果のフォーマットは下記の例の通り．

逐次音声認識結果


```
{
    recognition_id: "1e4a647d-636c-4d01-a068-d3cd7f1cc206",
    result_status: 2,
    sentences: [
        {
            converter_result: " 何が ",
            score: 1,
            words: [
                {
                    end_time: 44,
                    label: " 何 ; 何 ; ナニ ; ナニ ; 代 :::;: 何 ",
                    score: -856.60070800000005,
                    start_time: 31
                },
                {
                    end_time: 58,
                    label: " が ; が ; ガ ; ガ ; 助格 :::;: が ",
                    score: -304.80239899999998,
                    start_time: 58
                }
            ]
        }
    ]
}
```


最終音声認識結果


```
{
    recognition_id: "1e4a647d-636c-4d01-a068-d3cd7f1cc206",
    result_status: 0,
    sentences: [
        {
            converter_result: " 何が出来るの ",
            score: 0.68702399999999997,
            words: [
                {
                    end_time: 0.45000000000000001,
                    label: " 何 ; 何 ; ナニ ; ナニ ; 代 :::;: 何 ",
                    score: 0.88007599999999997,
                    start_time: 0.01
                },
                {
                    end_time: 0.58999999999999997,
                    label: " が ; が ; ガ ; ガ ; 助格 :::;: が ",
                    score: 0.880023,
                    start_time: 0.45000000000000001
                },
                {
                    end_time: 0.98999999999999999,
                    label: " 出来る ; 出来る ; デキル ; デキル ; 動非 : 上カ : 連体 :;: 出来る ",
```

```
                    score: 0.500413,
                    start_time: 0.58999999999999997
                },
                {
                    end_time: 1.1299999999999999,
                    label: " の ; の ; ノ ; ノ ; 助格 :::;: の ",
                    score: 0.52087099999999997,
                    start_time: 0.98999999999999999
                }
            ]
        },
        {
            converter_result: " 何ができるの ",
            score: 0.63773100000000005,
            words: [
                {
                    end_time: 0.45000000000000001,
                    label: " 何 ; 何 ; ナニ ; ナニ ; 代 :::;: 何 ",
                    score: 0.88007599999999997,
                    start_time: 0.01
                },
                {
                    end_time: 0.58999999999999997,
                    label: " が ; が ; ガ ; ガ ; 助格 :::;: が ",
                    score: 0.880023,
                    start_time: 0.45000000000000001
                },
                {
                    end_time: 0.98999999999999999,
                    label: " できる ; できる ; デキル ; デキル ; 動非 : 上カ : 連体 :;: できる ",
                    score: 0.37948799999999999,
                    start_time: 0.58999999999999997
                },
                {
                    end_time: 1.1299999999999999,
                    label: " の ; の ; ノ ; ノ ; 助格 :::;: の ",
                    score: 0.47840199999999999,
                    start_time: 0.98999999999999999
                }
            ]
        },
        {
            converter_result: " なにができるの ",
            score: 0.26099600000000001,
            words: [
                {
                    end_time: 0.45000000000000001,
                    label: " なに ; なに ; ナニ ; ナニ ; 代 :::;: なに ",
                    score: 0.118225,
                    start_time: 0.01
                },
                {
                    end_time: 0.58999999999999997,
```

```
                    label: " が ; が ; ガ ; ガ ; 助格 :::;: が ",
                    score: 0.11822199999999999,
                    start_time: 0.45000000000000001
                },
                {
                    end_time: 0.98999999999999999,
                    label: " できる ; できる ; デキル ; デキル ; 動非 : 上カ : 連体 :;: できる ",
                    score: 0.098805000000000004,
                    start_time: 0.58999999999999997
                },
                {
                    end_time: 1.1299999999999999,
                    label: " の ; の ; ノ ; ノ ; 助格 :::;: の ",
                    score: 0.47840199999999999,
                    start_time: 0.98999999999999999
                }
            ]
        }
    ]
}
```


recognition_id

SPF により発話単位で振られる UUID.

result_status

result_status は音声認識結果の種別を示す値. result_status が 0 の場合のみ，音声認識結果が NLU に送信される.

値	意味
0	最終認識結果
1	終話検知による認識結果
2	逐次認識結果
3	最大発話検知超過結果
281	認識結果なし

sentences

- converter_result
  NLU に送信される音声認識結果の文字列.
- score
  音声認識結果のスコア.
  result_status が 0 の場合，値は 0 〜 1 となる.
  result_status が 2 の場合，値は 1 となる.
- words
  - end_time
    認識エンジンに送られていた単語毎の音声終了時間.
    result_status が 0 の場合，値は 0 〜 10 (秒) となる.
    result_status が 2 の場合，値は 0 〜 1000 (フレーム) となる. ただし，1 フレーム= 10 ミリ秒とする.
    SPF やアプリで最大発話時間が変更された場合には，その値が最大となる.
  - label
    単語毎の音声認識結果文字列. 以下を ；区切りで設定する.

    * 単語の名称 (word_name)
    * 単語の表記 (surface)
    * 単語の仮名書き (kana)
    * 単語の口語読み (reading)
    * 品詞 (part_of_speech)
    * 活用語の標準形 (standard_form)
  - score
    単語毎の音声認識結果のスコア.
    result_status が 0 の場合，値は 0 〜 1 となる.
    result_status が 2 の場合，定義上 double 型の取り得る範囲の値となる.
  - start_time
    認識エンジンに送られていた単語毎の音声開始時間.
    result_status が 0 の場合，値は 0 〜 10 (秒) となる.
    result_status が 2 の場合，値は 0 〜 1000 (フレーム) となる．ただし，1 フレーム= 10 ミリ秒とする.
    SPF やアプリで最大発話時間が変更された場合には，その値が最大となる.

システム発話文字列の取得

下記のコードは，システム発話文字列を対話結果から取得するサンプルコードである.

```
private void OnMetaOut(string meta)
{
    //  システム発話文字列の取得
    string systemMessage = SpeechRecognitionJson.GetUtterance(meta);
}

[System.Serializable]
public class SpeechRecognitionJson
{
    public SystemText systemText;
    [System.Serializable]
    public class SystemText
    {
        public string utterance;
        public string expression;
    }
    public static String GetUtterance(string json)
    {
        SpeechRecognitionJson metaOutJson = JsonUtility.FromJson<SpeechRecognitionJson
    >(json);
        if ( metaOutJson.systemText != null &&
            metaOutJson.systemText.utterance != null &&
            metaOutJson.systemText.utterance != "")
        {
            //  システム発話文字列
            return metaOutJson.systemText.utterance;
        }
        else
        {
            return null;
        }
    }
```

```
}

void InitializeSDK()
{
    Speak.Instance().SetOnMetaOut(OnMetaOut);
}
```

7.3 メタ情報の受信

setOnMetaOut メソッドでイベントハンドラを設定する事で以下のメタ情報を受信できる．

返却されるメタ情報の種別一覧

メタ情報の種別は， type フィールドの値によって識別できる．ユーザ発話の開始を起点とした場合に返却されるメタ情報の種別を以下の表に示す．

type の値	メタ情報の種別	返却される条件
speech_recognition_vad_start	ユーザ発話の開始	必ず返却される
speech_recognition_vad_end	ユーザ発話の終了	必ず返却される
speech_recognition_result	音声認識結果	テキスト情報として返却される音声認識結果の result_status が 0 である場合
nlu_result	対話結果	テキスト情報として返却される音声認識結果の result_status が 0 である場合
error	エラー情報の通知	詳細な情報をユーザに通知すべきエラーが発生した場合

ユーザ発話の開始

ユーザ発話の開始を検知した際に返却されるメタ情報．フォーマットは下記の通り．

```
{
  "version": "",
  "type": "speech_recognition_vad_start"
}
```

ユーザ発話の終了

ユーザ発話の終了を検知した際に返却されるメタ情報．フォーマットは下記の通り．

```
{
  "version": "",
  "type": "speech_recognition_vad_end"
}
```

音声認識結果

テキスト情報として返却される音声認識結果と異なり，メタ情報として返却される音声認識結果は，音声認識結果が NLU に送信された場合のみ返却される．メタ情報として返却される音声認識結果のフォーマットは下記の例の通り．speechrec_result 以下はテキスト情報として返却される音声認識結果と同一フォーマットである．


```
{
    speechrec_result: {
        recognition_id: "1e4a647d-636c-4d01-a068-d3cd7f1cc206",
        result_status: 0,
        sentences: [
            {
                converter_result: " 何が出来るの ",
                score: 0.68702399999999997,
                voiceText: " 何が出来るの ",
                words: [
                    {
                        end_time: 0.45000000000000001,
                        label: " 何 ; 何 ; ナニ ; ナニ ; 代 :::;: 何 ",
                        score: 0.88007599999999997,
                        start_time: 0.01
                    },
                    {
                        end_time: 0.58999999999999997,
                        label: " が ; が ; ガ ; ガ ; 助格 :::;: が ",
                        score: 0.880023,
                        start_time: 0.45000000000000001
                    },
                    {
                        end_time: 0.98999999999999999,
                        label: " 出来る ; 出来る ; デキル ; デキル ; 動非 : 上力 : 連体 :;: 出来
    る ",
                        score: 0.500413,
                        start_time: 0.58999999999999997
                    },
                    {
                        end_time: 1.1299999999999999,
                        label: " の ; の ; ノ ; ノ ; 助格 :::;: の ",
                        score: 0.52087099999999997,
                        start_time: 0.98999999999999999
                    }
                ]
            },
            {
                converter_result: " 何ができるの ",
                score: 0.63773100000000005,
                voiceText: " 何ができるの ",
                words: [
                    {
                        end_time: 0.45000000000000001,
                        label: " 何 ; 何 ; ナニ ; ナニ ; 代 :::;: 何 ",
```

```
                    score: 0.88007599999999997,
                    start_time: 0.01
                },
                {
                    end_time: 0.58999999999999997,
                    label: " が ; が ; ガ ; ガ ; 助格 :::;: が ",
                    score: 0.880023,
                    start_time: 0.45000000000000001
                },
                {
                    end_time: 0.98999999999999999,
                    label: " できる ; できる ; デキル ; デキル ; 動非 : 上カ : 連体 :;: できる ",
                    score: 0.37948799999999999,
                    start_time: 0.58999999999999997
                },
                {
                    end_time: 1.1299999999999999,
                    label: " の ; の ; ノ ; ノ ; 助格 :::;: の ",
                    score: 0.47840199999999999,
                    start_time: 0.98999999999999999
                }
            ]
        },
        {
            converter_result: " なにができるの ",
            score: 0.26099600000000001,
            voiceText: " なにができるの ",
            words: [
                {
                    end_time: 0.45000000000000001,
                    label: " なに ; なに ; ナニ ; ナニ ; 代 :::;: なに ",
                    score: 0.118225,
                    start_time: 0.01
                },
                {
                    end_time: 0.58999999999999997,
                    label: " が ; が ; ガ ; ガ ; 助格 :::;: が ",
                    score: 0.11822199999999999,
                    start_time: 0.45000000000000001
                },
                {
                    end_time: 0.98999999999999999,
                    label: " できる ; できる ; デキル ; デキル ; 動非 : 上カ : 連体 :;: できる ",
                    score: 0.098805000000000004,
                    start_time: 0.58999999999999997
                },
                {
                    end_time: 1.1299999999999999,
                    label: " の ; の ; ノ ; ノ ; 助格 :::;: の ",
                    score: 0.47840199999999999,
                    start_time: 0.98999999999999999
                }
```

```
                ]
            }
        ]
    },
    type: "speech_recognition_result",
    version: ""
}
```

version

バージョン情報．ただし，現在は未使用となっている．

type

メタ情報の種別を識別するための文字列．
固定値として speech_recognition_result が入れられる．

speechrec_result

- recognition_id
  SPF により発話単位で振られる UUID．
- result_status
  result_status は音声認識結果の種別を示す値．
  メタ情報として返却される音声認識結果においては，0 が入れられる．
- sentences
  - converter_result
    音声認識結果の文字列．
  - score
    音声認識結果のスコア．
    値は 0 〜 1 となる．
  - voice_Text
    音声認識結果文字列．
  - words
    * score
      単語毎の音声認識結果のスコア．
      値は 0 〜 1 となる．
    * label
      単語毎の音声認識結果文字列．以下を;区切りで設定する．
      · 単語の名称（word_name）
      · 単語の表記（surface）
      · 単語の仮名書き（kana）
      · 単語の口語読み（reading）
      · 品詞（part_of_speech）
      · 活用語の標準形（standard_form）
    * start_time
      認識エンジンに送られていた単語毎の音声開始時間．
      値は 0 〜 10（秒）となる．
      SPF やアプリで最大発話時間が変更された場合には，その値が最大となる．
    * end_time
      認識エンジンに送られていた単語毎の音声終了時間．
      値は 0 〜 10（秒）となる．
      SPF やアプリで最大発話時間が変更された場合には，その値が最大となる．

対話結果

対話結果として返却されるメタ情報の仕様はこちらを参照のこと.

エラー情報の通知

詳細なエラー情報をユーザに通知する際に返却されるメタ情報. フォーマットは下記の通り.

```
{
  "type": "error",
  "systemText": {
    "expression": "NLU:Format of clientData is invalid."
  },
  "cause": {
    "invalidData": "{\"cacheFlag\":false,\"clientData\":{\"foo\":\"bar\"},\"voiceText
    \":\"#PB\"}"
  }
}
```

expression

エラーメッセージ.

cause

エラー原因.
データ不正による場合は invalidData フィールドに当該データが格納される.

ユーザ発話文字列の取得

下記のコードはメタ情報として返却された音声認識結果からユーザ発話文字列を取得するサンプルコードである.

```
public void OnMetaOut(string mateText)
{
    var speecMetaData = SpeechRecognitionJson.CreateFromJSON(mateText);
    //  ユーザ発話文字列取得
    string userMessage = MetaFindVoiceText(speecMetaData.speechrec_result);
}

private string MetaFindVoiceText(SpeechRecognitionJson speechrec)
{
    if(speechrec.sentences != null)
    {
        foreach (SpeechRecognitionJson.Sentence sentence in speechrec.sentences)
        {
            if (sentence.voiceText != null && sentence.voiceText != "")
            {
                //  ユーザ発話文字列
                return sentence.voiceText;
```

```
            }
        }
    }
    return null;
}

[System.Serializable]
public class SpeechRecognitionJson
{
    public SpeechrecResult speechrec_result;
    [System.Serializable]
    public class Sentence
    {
        public string converter_result;
        public string voiceText;
    }

    [System.Serializable]
    public class SpeechrecResult
    {
        public List<Sentence> sentences;
    }

    public static SpeechRecognitionJson CreateFromJSON(string json)
    {
        return JsonUtility.FromJson<SpeechRecognitionJson>(json);
    }
}

void InitializeSDK()
{
    Speak.Instance().SetOnMetaOut(OnMetaOut);
}
```

第 8 章

SDK の高度な制御

8.1 テキスト情報による対話

Speak#PutMeta メソッドを使用することで，任意のメタ情報での自然文による対話が可能である．以下のサンプルコードのように，voiceText に自然文を，その他のプロパティにメタ情報を設定して Speak#PutMeta メソッドを実行することで任意のメタ情報での自然文による対話ができる．

```
Speak.Instance().PutMeta("{ \"voiceText\" : \" こんにち
    は \" , \"language\":\"ja-JP\" }");
```

8.2 音声入出力の制御

音声入力の ON/OFF

音声入力を OFF にしたい場合は Speak#Mute メソッドを実行する． ON に戻したい場合は Speak#Unmute メソッドを実行する．

```
//  音声入力を OFF
Speak.Instance().Mute();

//  音声入力を ON
Speak.Instance().Unmute();
```

合成音声の再生のキャンセル

システム発話文字列の音声データの再生は途中でキャンセルすることが可能である．キャンセルを行う場合は，Speak#CancelPlay メソッドを使用する．

```
Speak.Instance().CancelPlay();
```

マイクミュート状態での起動

マイクミュート状態で接続を開始する場合には, 停止状態時に以下のようなパラメータ設定を行う．

```
Speak.Instance().SetMicMute(true);
```

8.3　ログ出力の設定

不具合発生時の問い合わせにおいて，事象調査のために詳細なログが必要となる場合がある．停止状態時に以下のようなパラメータ設定を行うことで，デバッグ情報に関するログを出力することができる．

```
Speak.Instance().Set("DisableLog", false);
Speak.Instance().Set("LogLevel", "DEBUG");
```

第 9 章

注意事項

9.1 サーバと接続中にアプリケーションがバックグラウンドに移行した際の注意事項

アプリケーションがバックグラウンドに移行した場合は，以下のサンプルコードのように接続の停止を行い，フォアグラウンドに移行した時に接続を再開すること． Unity の Standalone をプラットフォームに設定した場合は下記を ON に設定する必要がある．
PlayerSettings > Resolution and Presentation > Resolution > Run In Backgroud

```
void OnApplicationPause (bool pauseStatus) {
    if (pauseStatus)
    {
        Speak.Instance().Stop(OnStop);
        while (Speak.Instance().Poll(true)) { }
    }
    else
    {
        Speak.Instance().Start(OnStart, OnFailed);
    }
}
```

9.2 SDK を実行するスレッドに関する注意事項

SDK に対する操作を実行するスレッド

SDK に対する各操作はメインスレッドから実行する必要がある．メインスレッドから実行すべき操作は下記の通り．

- Speak#Start
- Speak#Stop
- Speak#PutMeta
- Speak#Mute
- Speak#Unmute
- Speak#CancelPlay

メインスレッド以外のスレッドから上記の操作が必要となった場合は，以下のサンプルコードのように .NET の SyncronizationContext を使用する．

```
using System.Threading;

private SynchronizationContext context;
private Speak speak;
```

```
void Start() {
    context = SynchronizationContext.Current;
    speak = Speak.Instance();
}

void Sample() {
    context.Post(__ =>
    {
        //  メインスレッドで実施する処理を記述する
        speak.Start(OnStart, OnFailed);
    }, null);
}
```

SDK に設定したイベントハンドラが実行されるスレッド

以下のメソッドおよびプロパティにて SDK に設定された各イベントハンドラはメインスレッド上で実行される．

- Speak#Start
- Speak#OnStart
- Speak#OnFailed
- Speak#Stop
- Speak#OnStop
- Speak#SetOnPlayStart
- Speak#OnPlayStart
- Speak#SetOnPlayEnd
- Speak#OnPlayEnd
- Speak#SetOnTextOut
- Speak#OnTextOut
- Speak#SetOnMetaOut
- Speak#OnMetaOut

9.3　SDK の自動停止に関する注意事項

対話が行われていない間は SDK を停止することが推奨される． そのため，SDK の開始または対話の終了から一定時間経過しても次の対話が開始されない場合は，SDK を自動停止するような処理が必要となる．

対話の開始と終了の検知

以下のイベント発生時に対話が開始される．

- 音声による対話: メタ情報として音声認識結果が返却された時
- テキスト情報による対話: Speak#PutMeta が実行された時

以下のイベント発生時に対話が終了する．

- メタ情報として返却された対話結果に utterance が含まれない時
- OnPlayEnd のイベントハンドラがコールバックされた時

下記のコードは，上記の対話の開始と終了の検知方法を利用して， SDK の開始または対話の終了から一定時間経過しても，次の対話が開始されない場合に SDK を停止するサンプルコードである．

```
using System.Threading;

public class SpeakSDKManager : MonoBehaviour
{
    Speak Speak.Instance();
    SynchronizationContext mContext;
    private float TIMEOUT = 10.000f;//10000/ms
    private int mDialogCunter = 0;
    void Start()
    {
        Speak.Instance() = Speak.Instance();
        mContext =  SynchronizationContext.Current;
    }

    public void StartSpeakSDK()
    {
        Speak.Instance().Start(OnStart, OnFailed);
    }

    public void PutMeta(string message)
    {
        Speak.Instance().PutMeta(json);
        CancelInvoke("AutoStopTask");
        DialogCunterIncrementAndGet();
    }

    private void OnStart()
    {
        Invoke("AutoStopTask", TIMEOUT);
    }

    private void OnStop()
    {
        //  接続停止完了時に実行する処理を記述
    }

    private void OnFailed(int ecode, string failstr)
    {
        //  接続失敗時に実行する処理を記述
    }

    private void OnMetaOut(string meta)
    {
        if(OnMetaOutJson.GetType(meta) == "speech_recognition_result")
        {
            //  音声対話開始
            CancelInvoke("AutoStopTask");
        }
        else if(OnMetaOutJson.IsUtterance(meta))
        {
            if(OnMetaOutJson.GetType(meta) == "nlu_result" &&
                DialogCunterDecrementAndGet() == 0)
```

```
            {
                //  音声対話終了
                Invoke("AutoStopTask", TIMEOUT);
            }
        }
    }

    private void OnPlayEnd(string text)
    {
        CancelInvoke("AutoStopTask");
    }

    private int DialogCunterIncrementAndGet()
    {
        mDialogCunter ++;
        return mDialogCunter;
    }

     private int DialogCunterDecrementAndGet()
     {
        mDialogCunter --;
        return mDialogCunter;
    }

    private void AutoStopTask()
    {
         ThreadPool.QueueUserWorkItem((status)=>
        {
            mContext.Post(__ =>
            {
                //  メインスレッドで実施する処理を記述する
                Speak.Instance().Stop(OnStop);
            }, null);
        });
    }

    [System.Serializable]
    public class OnMetaOutJson
    {
        public SystemText systemText;
        public string type;
        [System.Serializable]
        public class SystemText
        {
            public string utterance;
            public string expression;
        }

        public static bool IsUtterance(string json)
        {
            OnMetaOutJson metaOutJson = JsonUtility.FromJson<OnMetaOutJson>(json);
            return metaOutJson.systemText !=null &&  metaOutJson.systemText.utterance
    != null && metaOutJson.systemText.utterance != "";
```

```
        }

        public static string GetType(string json)
        {
            OnMetaOutJson metaOutJson = JsonUtility.FromJson<OnMetaOutJson>(json);
            if(metaOutJson.type != null &&  metaOutJson.type != "")
            {
                return metaOutJson.type;
            }
            else{
                return null;
            }
        }
    }
}
```

第 10 章

トラブルシューティング

10.1 FAQ

対話開始時のトラブル

- `Speak#Start` 呼び出し後にアプリがすぐ終了してしまう
  アプリケーションの permission が正しく設定されているかを確認する.
  設定した URL が正しいか確認する.
  別のネットワークに接続するか，ネットワーク接続状況の確認を行う.

- `Speak#Start` 呼び出し後にタイムアウトで切断されてしまう
  以下のサンプルコードのようにタイムアウト設定値を延伸する.

```
speak.Set("TimeoutTimeHeartbeat",10000); //ms
```

  アプリ側で設定可能なタイムアウト値を下記に示す.
  1. サーバとの接続処理時に適用されるタイムアウト値

設定キー	意味	値型	単位	デフォルト値
TimeoutTimeDNSResolve	DNS 名前解決のタイムアウト	int	ミリ秒	5000
TimeoutTimeConnect	TCP 接続タイムアウト	int	ミリ秒	5000
TimeoutTimeSocketPostInit	TCP ソケット open 後処理の TLS ハンドシェイクタイムアウト	int	ミリ秒	5000
TimeoutTimeOpenHandshake	websocket プロトコルにおける open ハンドシェイク時のタイムアウト	int	ミリ秒	5000
TimeoutTimeAuthentication	Speak のプロトコルにおける UDS 認証応答時のタイムアウト	int	ミリ秒	5000

  2. サーバとの接続完了後に適用されるタイムアウト値

設定キー	意味	値型	単位	デフォルト値
IntervalTimeHeartbeat	ハートビートを送信する間隔	int	ミリ秒	2000
TimeoutTimeHeartbeat	ハートビートタイムアウト	int	ミリ秒	2000

  タイムアウト設定値を延伸しても改善しない場合，以下を試みる.
  - DNS 名前解決のタイムアウトの場合
    * 利用している DNS サーバに問題がある可能性があるため，DNS サーバをパブリック DNS に変更してみる.
    * 利用している DNS サーバが IPv6 に対応していない可能性があるため，IPv6 アドレスの名前解決を無効化して試す. `csharp speak.Set("EnableIPv6DNSQuery",false);`
  - その他のタイムアウトの場合
    * 接続しているネットワーク内で遅延が発生している可能性があるため，別のネットワークに接続してみる.

- `Speak#Start` 呼び出し後，“UDS:Invalid Signature” というエラーメッセージが出て切断されてしまう
  設定したデバイストークンが正しいか確認する．
- `Speak#Start` 呼び出し後，“UDS:Signature Expired” というエラーメッセージが出て切断されてしまう
  デバイストークンの有効期限の延長処理を行う．
  デバイストークンのリフレッシュを行う．

音声認識のトラブル

- 話しかけても音声が認識されず，応答が返ってこない
  自然な口調ではっきりと話しかけてみる．
  `Speak#Mute`メソッドによって音声入力が OFF の状態になっていないか確認する．
  マイクミュート状態で接続を開始していないか確認する．

音声再生のトラブル

- 音声再生時に音飛びが発生する
  アプリケーション側で多量にスレッドを立てていないか確認する．
- 音声再生開始直後にエラーが発生する
  長い音声を再生する場合, SDK 内部の音声バッファが溢れてエラーとなることがある．以下の設定値でバッファサイズを変更すると改善する可能性がある．

設定キー	意味	値型	単位	デフォルト値
OggBufferSizeSamples	音声バッファサイズ	int	サンプル数	1048576 (約 47 秒の音声に相当)

対話実行中のトラブル

- アプリとの対話中に，対話が急に停止してしまった
  Speak クラスのインスタンスを複数生成していないか確認する．
- `Speak#Start` 呼び出し後，一定時間経過後にサーバ側から突然接続を切断された
  一定時間ユーザアクションがない場合，サーバ側から接続を切断する仕様となっている．そのため，SDK の開始または対話の終了から一定時間経過しても次の対話が開始されない場合は，SDK を自動停止するような処理が必要となる． 詳細は SDK の自動停止に関する注意事項を参照のこと．
- 一度バックグラウンドに遷移し，その後フォアグラウンドへと復帰した際，対話が中断されてしまっていた
  SDK の仕様として，アプリケーションがバックグラウンドに移行した場合には接続の停止を行い，フォアグラウンドに移行した時に接続を再開するような処理が必要となっている．詳細はサーバと接続中にアプリケーションがバックグラウンドに移行した際の注意事項を参照のこと．
- 設定したイベントハンドラ内で UI を操作する処理を実行した際，アプリケーションが落ちてしまった
  いくつかの API は UI スレッドから実行する必要がある． 詳細は SDK を実行するスレッドに関する注意事項を参照のこと．

対話終了時のトラブル

- `Speak#Stop` 呼び出し後，接続終了時に実行されるように設定したイベントハンドラが呼ばれない
  「停止状態」以外の状態でエラーが発生した場合，OnStart または OnStop イベントがアプリに通知されるとは限らない．そのため，OnFailed イベントが通知されていないかを確認する．

エラーコード一覧に記載のないエラーが発生した場合

- 接続先のサーバが一時的に使用不可となっている可能性があるため，いったんアプリケーションを再起動してから対話開始する．
- ネットワークが原因の可能性があるため，別のネットワークに接続するか，ネットワーク接続状況の確認を行う．
- 上記の対処を試しても事象が改善しない場合，担当者へ問い合わせる．

10.2 エラーコード

クライアントに通知されるエラーを下記に示す．記載外のエラーコードについては，FAQ の「エラーコード一覧に記載のないエラーが発生した場合」を参照のこと．

エラーコード	発生原因	対応
131	音声再生時に内部音声バッファが溢れた	音声再生のトラブルを参考に，設定値 `OggBufferSizeSamples` を増やして音声バッファサイズを拡張する
135	クライアント側で投入された PCM データのエンコード時に異常が発生した	アプリ側で PCM データを録音し投入している場合，データ内容を確認する
141	不正な設定値が設定された	設定値の確認を行う
146	Edge のキューサイズがデータ投入速度に対して不足している	Egde のキューサイズを拡張する
400	クライアントのリクエストが不正	SSL 接続ホストに対し，非 SSL 接続を行おうとしていないか確認する
404	存在しない URL に対してリクエストが行われた	設定した URL が正しいか確認する
その他 4XX, 5XX	サーバから HTTP エラーが返却された	返却されたエラー値は HTTP レスポンスコードであるため，その内容に従う
611	サーバ認証タイムアウトが発生した	通信遅延が発生しているため，別のネットワークに接続するか，対話開始時のトラブルを参考に設定値 `TimeoutTimeAuthentication` を延伸する
612	サーバ認証が失敗した	デバイストークンが正しいか確認する
1011	ハートビートタイムアウトが発生した	通信遅延が発生しているため，別のネットワークに接続するか，対話開始時のトラブルを参考に設定値 `TimeoutTimeHeartbeat` および `IntervalTimeHeartbeat` を延伸する
4000	websocketpp での接続に失敗した	別のネットワークに接続するか，ネットワーク接続状況の確認を行う
4001	DNS 情報の獲得に失敗した	設定した URL におけるホスト名が正しいか確認する
4002	URL のフォーマットが不正	URL フォーマットが正しいか確認する
4003	DNS への接続においてタイムアウトが発生した	UDP ポート (53) が閉塞していないか確認する．通信遅延が発生しているため，別のネットワークに接続するか，対話開始時のトラブルを参考に設定値 `TimeoutTimeDNSResolve` を延伸する
4004	TCP の接続においてタイムアウトが発生した	通信遅延が発生しているため，別のネットワークに接続するか，対話開始時のトラブルを参考に設定値 `TimeoutTimeConnect` を延伸する
4005	TLS handshake timeout が発生した	通信遅延が発生しているため，別のネットワークに接続するか，対話開始時のトラブルを参考に設定値 `TimeoutTimeSocketPostInit` を延伸する
4006	Websocket プロトコルの handshake がタイムアウトした	通信遅延が発生しているため，別のネットワークに接続するか，対話開始時のトラブルを参考に設定値 `TimeoutTimeOpenHandshake` を延伸する
その他 4XXX	Websocket エラーが発生した	返却された Websocket のエラーメッセージの内容に従う
40021	JSON 形式でない文字列をメタ情報として送信した	送信するメタ情報を JSON 形式に整形する

エラーコード	発生原因	対応
40101	クライアント側から指定されたデバイストークンが無効	設定したデバイストークンが正しいか確認する
40102	クライアント側から指定されたデバイストークンの有効期限が切れている	デバイストークンの有効期限の延長処理を行う
40103	サーバ側でクライアントからの認証要求タイムアウトが発生した	通信遅延が発生している可能性があるため，別のネットワークでの接続を試す
40104	サーバ側でクライアントから複数回の認証要求を受け取った	クライアント側で認証要求を複数回送信していないか確認する
40801	サーバ側でクライアントからの設定情報受信タイムアウトが発生した	通信遅延が発生している可能性があるため，別のネットワークでの接続を試す
50030	対話シナリオのシステム発話内容に，長い文字列や合成音に変換できない特殊な文字が含まれている	クライアント側で指定している対話シナリオの内容を再確認する
60133	サーバ側でのデータ流量制限に抵触した	アプリ側で音声データをストリーミングではなく一括して送信している場合，データ投入速度を落とす
60144	サーバ側でプロキシプロトコル不正データを検出した	クライアント側に定義されている Edge 名称がサーバ側と不一致になっていないか確認する
60148	サーバ側での通信速度制限に抵触した	クライアント側で不要なデータを短時間で大量送信していないか確認する
60149	サーバ側で一定時間クライアントからデータ受信がなかった	データを送信しない場合，クライアント側は自動切断するように実装する