全銀システムにおける

API ゲートウェイの検討に係る実証実験

結果報告書

一般社団法人全国銀行資金決済ネットワーク

2022 年 3 月

目次

1　本書の位置づけ

本報告書は、資金移動業者の全銀システムの参加に関する検討事項の一つである、API ゲートウェイの検討にあたって実施した実証実験の結果をとりまとめたものである。

2　検討経緯

昨年度、全銀ネットの「次世代資金決済システムに関する検討タスクフォース」における議論の中で、資金移動業者から、API による全銀システムへの接続を期待する意見があったこと等を踏まえ、同タスクフォースの報告書においても、「接続方式については、短期的には、現行システムを前提とした参加を協議しつつ、資金移動業者および既存加盟銀行の双方のメリットが期待できる API を活用した接続方法について、具体的な検討を進めることが望ましい。」との提言が盛り込まれた。

これを踏まえ本年度は、「次世代資金決済システムに関する検討タスクフォース」の傘下に設置したシステムワーキンググループにおいて、API 接続を前提とした新たな共通基盤（API ゲートウェイ）の仕様や必要機能等の具体的な検討を行うにあたり、標準的な API 接続におけるノックアウトファクターを早期に検証するとともに、API ゲートウェイの機能面および非機能面に係る検証等を行うため、実証実験を実施した。

3 実現に向けた論点整理

API ゲートウェイの実現にあたり、API ゲートウェイの稼動が加盟銀行の内国為替取引に係る業務・運用に影響を与えないよう、全銀システムとの接続は現行プロトコルを維持したうえで、参加者システムとの標準的・軽量な接続を実現することが求められる。つまり、API ゲートウェイと全銀システム間には現行プロトコルとしてすでに実現している接続方式を用いる一方、参加者システムと API ゲートウェイ間は、全銀システム側が現行プロトコルであることを前提に、新たに標準的・軽量な接続方式を実現する必要がある。

こうした前提を踏まえ、API ゲートウェイに DB を配置する等により、参加者システム側と全銀システム側のトランザクションを分け、双方の接続仕様の独立性を確保することが望ましいと整理した。そのうえで、現行の全銀システム側の接続仕様を前提にした API 仕様を策定し、参加者システムからの標準的・軽量な接続の実現性と、API ゲートウェイの実現方式について検証することとした。

本実証実験は、標準的な API 接続におけるノックアウトファクターを早期に検証することを目的としていることから、新たな接続方式の実現が必要な参加者システムと API ゲートウェイ間の接続を検証範囲とし、その実現性を確認した。

API ゲートウェイを介した全銀システムへの接続の実現性を確認するうえでは、為替電文を安定的に抜け漏れなく送受信できることが必須である。そして、それを実現する仕組みのなかでも、「通信方式」、「下り電文配信方式」、「冗脱管理（電文を抜け漏れ・重複なく処理する仕組み）」が、現行プロトコルと標準的・軽量な接続方式でギャップが大きい要素として挙げられ、これらの両立の実現性がノックアウトファクターとなる可能性がある。

そのため本実証実験では、これらの処理方式をノックアウトファクター候補として標準的・軽量な接続が実現可能か検証した。各ノックアウトファクター候補について、実現に向けて必要な対処と現行プロトコルでの接続方式、標準的・軽量な API 接続方式を図表 3.1 に示す。

図表 3.1 ノックアウトファクター候補一覧

ノックアウトファクター候補	API ゲートウェイの実現に必要な対処	現行プロトコルでの接続方式	標準的・軽量な API 接続方式
通信方式	異なるステータス管理方式の独立性を確保	パス（論理的な経路）単位に接続元・接続先間でコネクション状態を常時共有しながら電文送受信を行う。	HTTP で 1 件 1 件コネクションは独立して電文送受信を行う。
下り電文配信方式	異なる下り電文の配信方式への変換	パス単位での常時接続を前提に、全銀センタからリアルタイム配信を行う。	1 件ごとに独立した接続であることを前提に、受信側から電文を受信する（PULL）、もしくは送信側から電文を送信する（PUSH）。

ノックアウトファクター候補	API ゲートウェイの実現に必要な対処	現行プロトコルでの接続方式	標準的・軽量なAPI 接続方式
冗脱管理	異なる冗脱管理（電文を抜け漏れ・重複なく処理する仕組み）の独立性を確保	パス単位で経路上の追い越しが原則発生しないことを前提に、受信側で通番の連続性を管理することで、抜け漏れ・重複を検知する。	1 件ごとに独立した接続であることを前提に、電文の重複受信は受信側で検知する。一方、電文の抜け漏れは送信側が応答の受信状況により検知する。

4　実証実験の内容

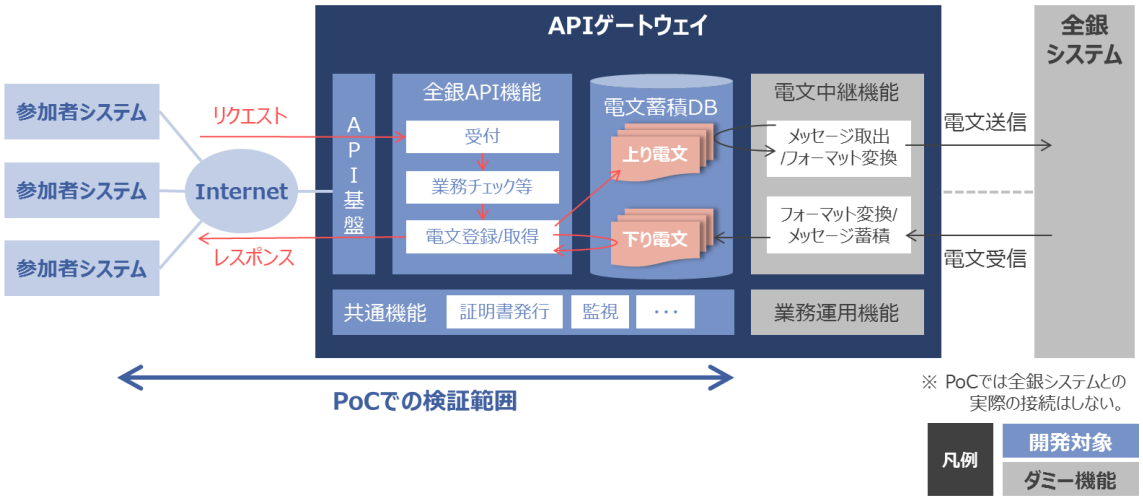
4.1　実施概要

実証実験の参加者は、NTT データが構築した実証実験向け API ゲートウェイに対し、参加者のシステムからインターネット経由でアクセスし、擬似的にテレ為替業務を実施した。また、実証実験における検証観点を踏まえ、参加者それぞれの検証を通じて得られた実施結果に対して評価を実施した。

実施にあたっては、図表 4.1 のとおり、クラウドサービス上の環境に必要な API を構築し、参加者から API でのテレ為替電文の送受信を可能とした。実証実験における検証対象は、フロントの API から API ゲートウェイ内の電文蓄積 DB までとした。検証に必要となる全銀システムとの接続に係る各種機能（テレ為替の擬似下り電文作成等）はダミー機能として提供した。

なお、今回採用する環境の技術基盤の検証は、本実証実験のスコープとしていない。

図表 4.1　システム概要図

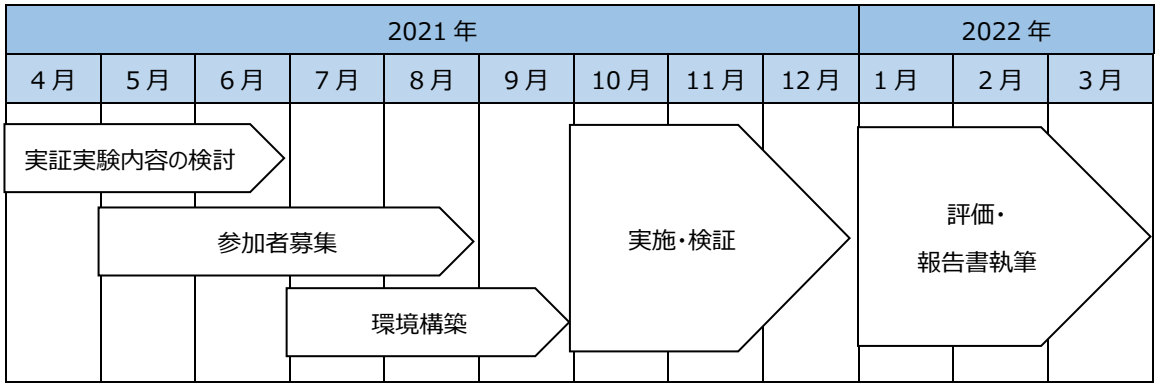


4.2　スケジュール・体制

(1)　スケジュール

2021 年 4 月から 2022 年 3 月までの活動期間において、図表 4.2 のスケジュールで実証実験を行った。

図表 4.2 実証実験スケジュール



(2)　実証実験の実施日と参加企業

実施・検証期間における、実証実験の実施日と参加企業を図表 4.3 に示す。

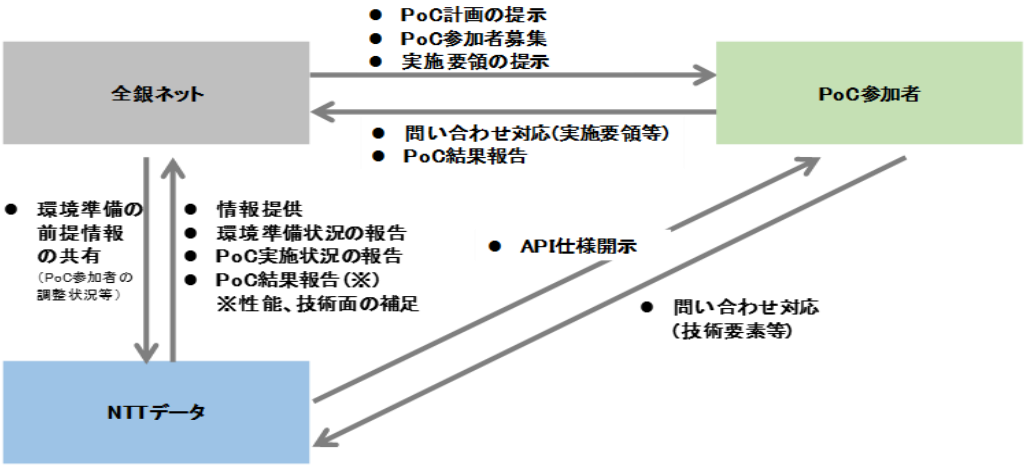
図表 4.3 実証実験の実施日と参加企業

	実施日	参加企業
1 回目	10 月 4 日（月）～10 月 8 日（金）	富士通株式会社
2 回目	11 月 1 日（月）～11 月 2 日（火）、 11 月 4 日（木）～11 月 5 日（金）	富士通株式会社 ワイズ・ペイメンツ・ジャパン株式会社
3 回目	11 月 8 日（月）～11 月 12 日（金）	富士通株式会社 ワイズ・ペイメンツ・ジャパン株式会社 GMO あおぞらネット銀行株式会社

(3) 実施体制と役割

実施体制と関係者ごとの役割を図表 4.4 に示す。

図表 4.4 実施体制と役割



4.3　本実証実験で採用した処理方式

参加者システムとの標準的・軽量な接続の実現に向けて、ノックアウトファクター候補とした処理方式である「通信方式」、「下り電文配信方式」、「冗脱管理」について、本実証実験での採用方式と採用理由について以下に示す（図表 4.5）。

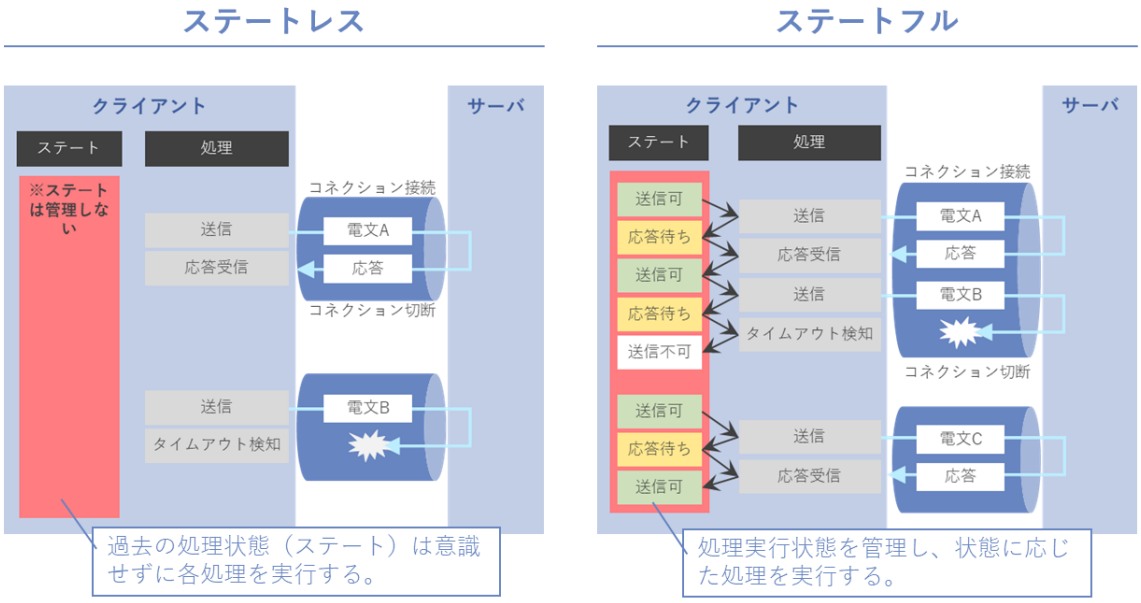
図表 4.5　本実証実験で採用した処理方式

項番	処理方式	本実証実験での採用方式
1	通信方式	1 件 1 件独立して扱えるステートレス方式
2	下り電文配信方式	参加者システムから配信要求を繰り返すポーリング方式
3	冗脱管理	送信側が電文毎に一意の ID を付与する ID 方式

(1)　通信方式

参加者システムと API ゲートウェイ間のステータスを管理する通信方式について、「ステートレス」と「ステートフル」が選択肢として挙げられる。ステートレスは、1 件 1 件の通信を独立したリクエスト・レスポンスとしてやり取りする方式で、代表技術としては REST や SOAP が挙げられ、web サービスでの利用や銀行分野のオープン API 標準として扱われている。一方、ステートフルは、通信相手と対話的に状態（ステート）を共有しながら連携する方式である。代表技術としては gRPC、WebSocket が挙げられ、スマホアプリや IoT 等のリアルタイムバックエンド通信に使用されることが多い。なお、現行全銀システムの接続方式は、ステートフルに該当する。ステートレスおよびステートフルのイメージ図を図表 4.6 に示す。

図表 4.6　ステートレスおよびステートフル



図表 4.7 のとおり、比較評価の結果、ステートフルの方が現行の全銀システムの方式に近く、通信の効率においても優位ではあるものの、早期に軽量で標準的な接続方式を実現するという API ゲートウェイの役割に鑑み、本実証実験では、現状の普及度やシステムの実装負担・スケーラビリティに優位性のある、ステートレスを採用した。

図表 4.7　ステートレスとステートフルの比較評価

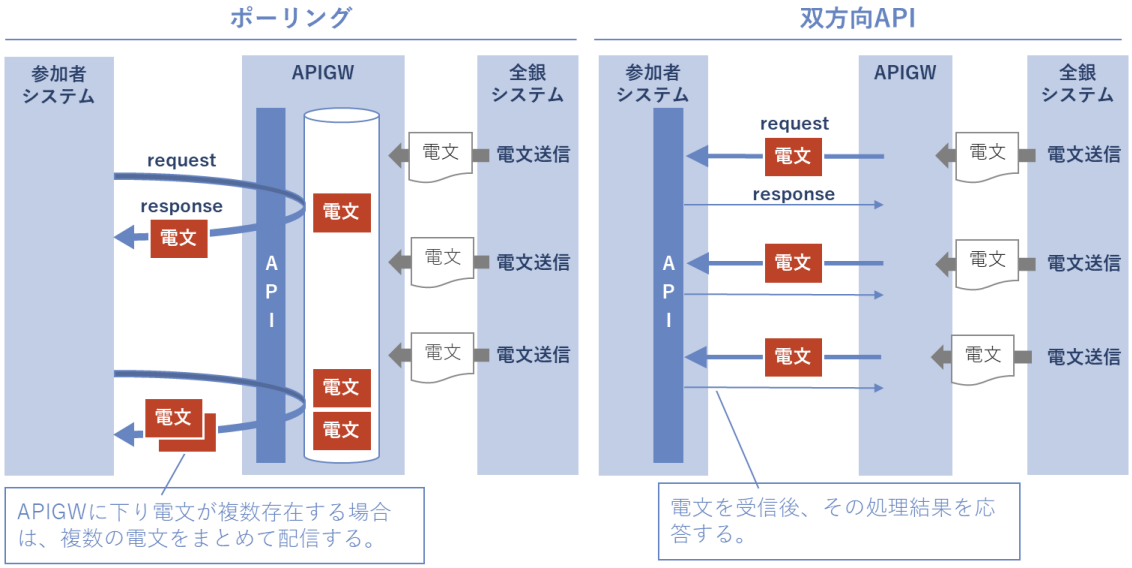
観点	ステートレス	ステートフル
システムの実装負担	○ 各通信が独立しているため、複雑度が低く、軽量な実装が可能。	× 状態の管理、状態遷移の制御が必要であるため、複雑度が高い。
水平拡張性（スケーラビリティ）	○ 同時に接続するコネクション数を動的に変更することが可能。	× コネクション数の拡張などは、ステートレスと比較すると複雑な対応になる。
通信量	△ 相対的には都度の通信量が増加するが、全銀システムの場合、電文のデータも大きくなく、影響は小さい。	○ 状態を前提に必要最小限の情報授受のみを行うため、通信量が少ない。
その他	－ 全銀協「オープン API のあり方に関する検討会報告書」の推奨方式。	－ 現行の全銀システムに近い方式。

(2)　下り電文配信方式

下り電文を配信する方式としては、参加者システム側から配信要求を繰り返し、その依頼に対して応答する形で下り電文を配信する「ポーリング」と、参加者システムが構築した API に対して API ゲートウェ

イからリクエストを送り下り電文を配信する「双方向 API」が選択肢として挙げられる。ポーリングおよび双方向 API のイメージ図を図表 4.8 に示す。

図表 4.8 ポーリングおよび双方向 API



図表 4.9 のとおり、本実証実験においては、参加者の API 構築負担を考慮し、短期的にはポーリングでの下り電文配信が現実的と考え、ポーリングを採用した。ただし、全銀システムと同様にリアルタイムでの電文受信の必要性や、中長期的な観点で、参加者システムとの同期的な処理（入金結果の同期等）を考慮していく場合には、双方向 API についても検討していく必要がある。

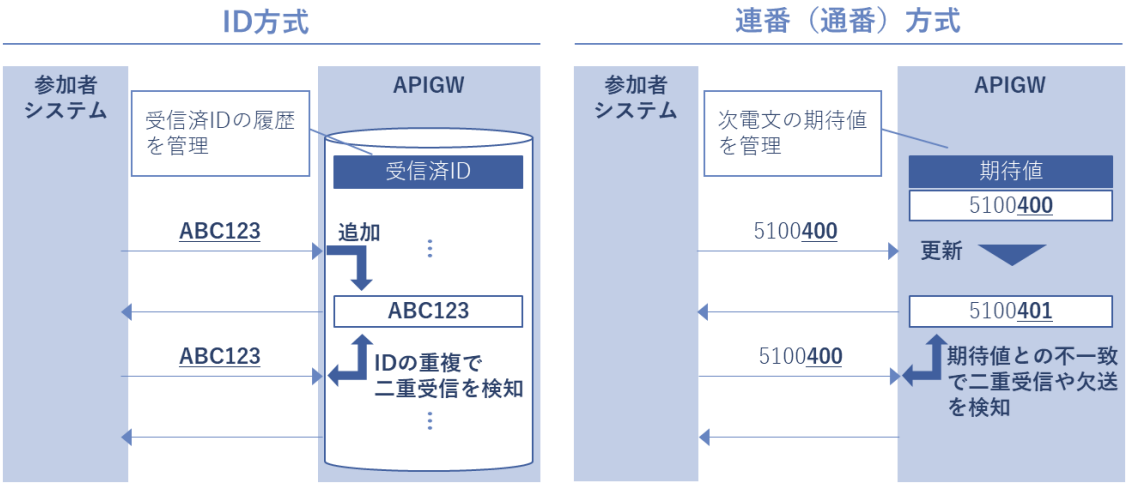
図表 4.9 ポーリングおよび双方向 API の比較評価

観点	ポーリング	双方向 API
開発範囲	○ APIGW で API のインタフェースを構築し、参加者システム側は応答を処理するのみ。	× 参加者システム側においても、API のインタフェースを構築する必要がある。
リアルタイム性	△ 間隔を空けて複数電文まとめて取得するためリアルタイム性は劣るが、取得依頼を並行させることにより、影響を抑えることが可能。	○ 下り電文発生毎に APIGW 側から配信処理を行うため、リアルタイムに配信可能。
(中長期) 同期処理	△ 下り電文の処理結果を APIGW 側で検知できないため、将来的に被仕向行まで含めた同期処理の実現を考える場合には適合しない。	○ 下り電文の処理結果を APIGW 側に応答してもらうことにより、同期処理の実現が可能。

(3) 冗脱管理

電文の漏れ、重複を検知する冗脱管理の仕組みとして、「ID 方式」と「連番（通番）方式」が選択肢として挙げられる。ID 方式は、参加者システムと API ゲートウェイ間での識別子として、送信側が電文ごとに一意の ID を付与し、受信側が受信済 ID との重複を確認し二重受信を検知する。欠送が発生した場合は、レスポンス未受信（タイムアウト）により送信側が検知する。一方、連番（通番）方式では、参加者システムと API ゲートウェイ間での識別子として、送信側が電文ごとに連番を付与し、受信側で期待値との不一致を確認することで、二重受信や欠送を検知する。また、欠送発生時は、タイムアウトにより送信側が検知することも可能である。ID 方式および連番（通番）方式のイメージ図を図表 4.10 に示す。

図表 4.10 ID 方式および連番（通番）方式



図表 4.11 のとおり、本実証実験では、通信方式にステートレスを採用してスケーラビリティや実装負担といった API 化のメリットを享受することに合わせ、冗脱管理においてもステートフルな連番（通番）方式ではなく、ID 方式を採用した。

図表 4.11 ID 方式および連番（通番）方式の比較評価

観点	ID 方式	連番（通番）方式	（参考） 全銀システムの方式
順序性	× 基本的に順序性は保証されない。応答の受信を確認しながら順次処理することも可能。	△ 複数コネクションで並行送信する場合、コネクション間の順序性は保証されない。	複数コネクションで並行送信する場合、コネクション間の順序性は保証されない。
スケーラビリティ	○ ID は並行付与可能であるため、スケールアウトが容易。	△ 統一的な連番付与が必要であり、スケールアウト対応がやや複雑。	統一的な連番付与が必要であり、スケールアウト対応がやや複雑。

観点	ID 方式	連番（通番）方式	（参考） 全銀システムの方式
変更容易性	○ 一意であればよいため、送信側の都合で体系の変更も可能。	△ 送信側と受信側間における期待値の整合性をとって体系等を変更する必要がある。	送信側と受信側間における期待値の整合性をとって体系等を変更する必要がある。
通信方式との親和性	○ ステートレス／ステートフルどちらの方式でも採用可能。	△ コネクションごと等でステート（期待通番）を管理。	コネクションごと等でステート（期待通番）を管理。

なお、通信方式、下り電文配信方式、冗脱管理の採用案の組み合わせの相性の評価を図表 4.12 に示す。本実証実験では、項番 1 の組み合わせである「ステートレス」、「ポーリング」、「ID 方式」を採用した。

図表 4.12　通信方式、下り電文配信方式、冗脱管理の組み合わせ評価

項番	通信方式	下り電文配信	冗脱管理	評価	
1	ステートレス	ポーリング	ID	◎	採用案それぞれに優位性があり、今回採用を想定している組み合わせ。また、組み合わせの相性も良い。
2			連番	×	ステート（期待通番）を管理する通番方式は、ステートレスにそぐわず、採用は推奨されない。
3		双方向 API	ID	△	方式の組み合わせは採りうるが、双方向 API では、参加者システム側での API インタフェースの構築が必要になる。
4			連番	×	参加者での API 構築負担に加え、ステート（期待通番）を管理する通番方式は、ステートレスにそぐわず、採用は推奨されない。
5	ステートフル	ポーリング	ID	×	ステートフルの特徴であるリアルタイム性が、ポーリングの随時性により損なわれ、組み合わせの相性が良くない。
6			連番	×	項番 5 と同様。
7		双方向 API	ID	△	項番 3 と同様。
8			連番	△	項番 3 と同様。

4.4　検証観点

図表 4.13 のとおり、ノックアウトファクターの検証に必要となる「基本形」、「冗脱管理／対処」、「処理性能」の 3 項目を検証観点とした。

「基本系」、「冗脱管理／対処」は、機能面において、ノックアウトファクター候補となる処理方式（ステートレス方式／ポーリング方式／ID 方式）でのテレ為替業務が実施可能か確認する目的で、API ゲートウェイおよび参加者システム双方の目線で検証を行った。

「処理性能」は、採用した処理方式における性能への影響を確認する目的で検証を行った。性能評価として基本的な検証観点である「スループット」、「レスポンスタイム」のほか、API ゲートウェイでは電文蓄積 DB を配置する構成を想定していることから、「キャパシティ要件における性能への影響」について検証を行った。また、現行仕様と大きく異なる点として、下り電文配信方式としてポーリングを採用したことより、ポーリングの実施条件によって API ゲートウェイに電文が滞留する時間が発生する可能性があることから、「ポーリングによる仕向から被仕向への電文到達時間への影響」について検証を行った。

図表 4.13　検証観点

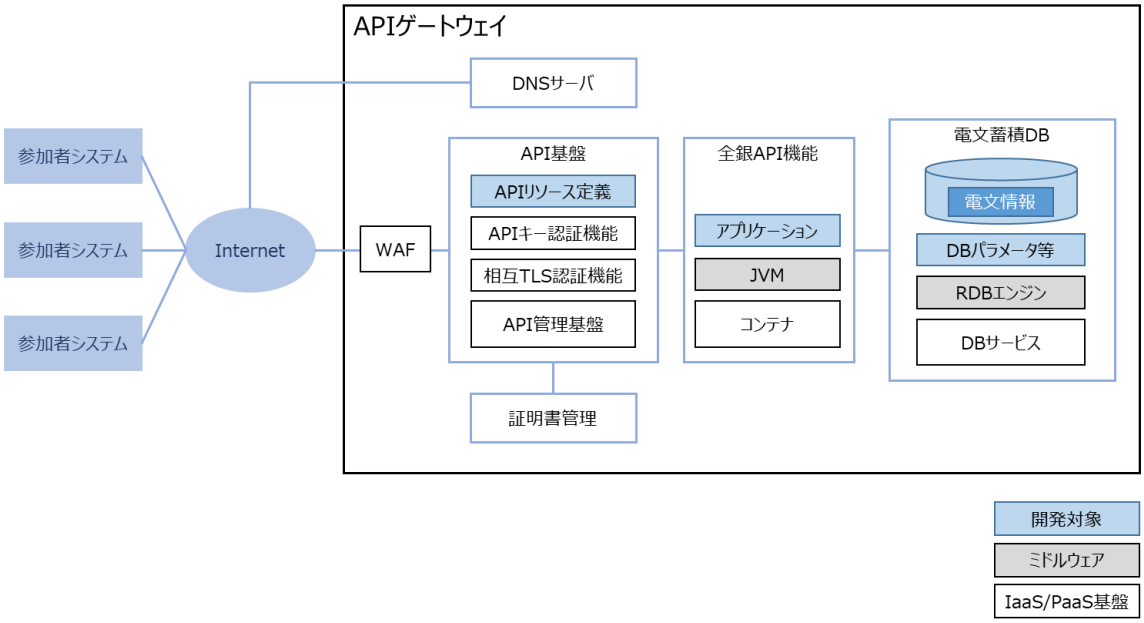
項目		検証主体	検証観点
基本系	仕向観点	全銀ネット 参加者 NTT データ	・参加者システムから上り電文の送信、API ゲートウェイからの処理結果の受信が API 仕様書どおり実現できること
	被仕向観点		・参加者システムからの下り電文配信依頼（ポーリング）、API ゲートウェイからの下り電文の配信が API 仕様どおり実現できること ・1 回のポーリングで配信された複数の下り電文を参加者システムで処理（入金等）できること
冗脱管理／対処		全銀ネット 参加者 NTT データ	・ ユニークな ID が付与された電文それぞれに対して、API ゲートウェイ／参加者システムにおいて、抜け漏れなく処理できること ・ ID かつ内容が重複した上り電文を受信した場合、API ゲートウェイで二重処理しないこと、また、ID が重複した下り電文を再取得した場合、参加者システムにおいて二重処理しないこと ・ 上り電文やポーリングの結果が一定時間以上返却されない場合、参加者システムでタイムアウトを検知し、再送できること、また、再送電文が正常に処理されること
処理 性能	接続先単位および API ゲートウェイ全体のスループット	全銀ネット NTT データ	・ 現行の加盟銀行の単位時間あたり取引量等を踏まえた負荷モデルを用いて確認する ・ データアクセス競合等によりスケールアウトでは

項目		検証主体	検証観点
	各 API のレスポンスタイム		解決できないボトルネックが存在しないか確認する ・ 各 API の処理性能が、処理特性（参照系・更新系等）と大幅に乖離が無いか確認する
	キャパシティ要件の性能への影響		・ 現行の加盟銀行の 1 日あたり取引量と電文保管要件を踏まえた格納データ量で測定する ・ データ蓄積時も大幅な性能劣化が無いことを確認する
	ポーリングによる仕向から被仕向への電文到達時間への影響		・ ポーリング 1 回あたりの取得電文数を変化させた場合における処理時間、取得効率、システム負荷への影響を確認する

4.5　環境

本実証実験は、図表 4.14 のとおり、クラウドサービス上に環境を構築した。

図表 4.14 環境構成図



4.6 実装機能

(1) API 一覧

本実証実験を行うにあたり、ノックアウトファクターの検証に必要な機能として、図表 4.15 のとおり API を実装した。

図表 4.15 API 一覧

項番	機能	メソッド	概要
1	上り電文送信	POST	参加者から送信された電文を受信後、電文内容に対するチェックを実施し、問題が無ければ、電文を DB に登録後、送信元に正常応答を返却する。エラーを検知した場合は、エラー応答を返却する。なお、受信したリクエスト ID がすでに使用済みであった場合は、DB への登録処理は行わずに、当該リクエスト ID をキーとして、DB に登録済みの電文との比較を行う。内容が一致した場合には、登録時と同内容の応答を返却する。内容が不一致の場合は、エラー応答を返却する。
2	上り電文照会	GET	参加者から送信された照会依頼を受信後、参加者から当日に受信したすべての上り電文の電文情報（リクエスト ID、上り電文処理状況）を通知する。 照会依頼時に上り電文処理状況（未処理／処理済／エラー）を指定した場合は、指定値に紐づく上り電文の電文情報のみを通知する。
3	下り電文配信要求	GET	参加者からの下り電文配信要求を受信後、未配信の下り電文を配信する。この際、配信対象となった下り電文に対し、配信要求のリクエスト ID を紐づけて保管する。なお、リクエスト ID がすでに使用済みであった場合には、当該リクエスト ID で配信済みの下り電文を再度配信する。
4	下り電文照会	GET	参加者から送信された照会依頼を受信後、APIGW が保有している当日に受信したすべての下り電文の電文情報（リクエスト ID、電文 ID、下り電文処理状況）を通知する。 照会依頼時に下り電文処理状況（未配信／配信済）を指定した場合は、指定値に紐づく下り電文の電文情報のみを通知する。

(2) 電文フォーマット

電文フォーマットを定義するうえで、データ表現形式（データの定義方法）および電文項目（電文データへの意味付け方法）について、本実証実験における採用方針を検討し、データ表現形式は「JSON」、電文項目は「電文種目等ごとに定義」する方針とした。採用理由について以下に示す。

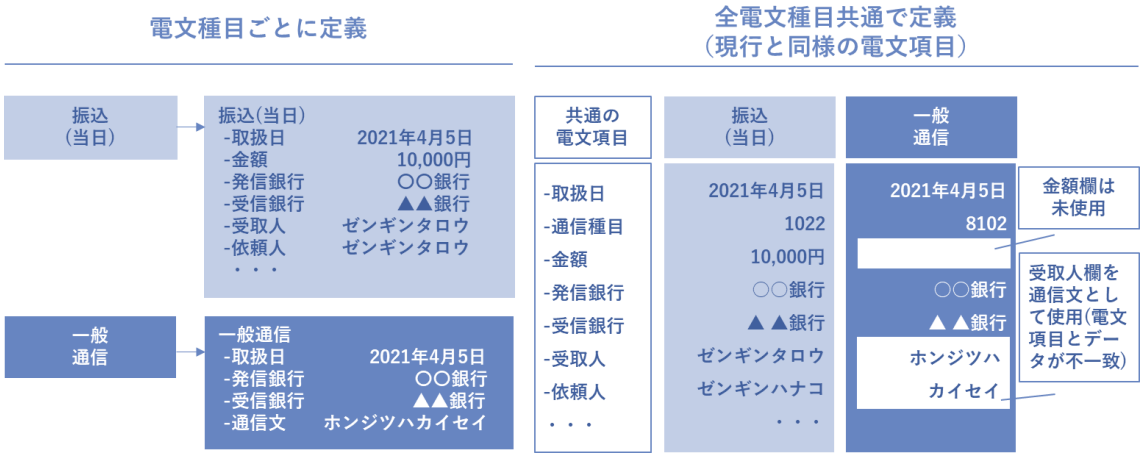
データ表現形式については、一般に広く標準とされている「JSON」形式および「XML」形式が選択肢として挙げられる。図表 4.16 のとおり、比較評価の結果、本実証実験では、通信データ量や処理性能の評価が比較的高く、かつ参加者システムにおいて開発容易性の高い（オープン API として開発標準と認められており、世の中に広く採用されている）JSON を採用した。

図表 4.16 JSON および XML の比較評価

観点	JSON	XML	（参考）帳票形式
柔軟性	○ 回数不定の繰り返し等への対応も可能であり、スキーマという定義ファイルを活用することで柔軟な表現が可能。	○ 回数不定の繰り返し等への対応も可能であり、スキーマという定義ファイルを活用することで柔軟な表現が可能。	回数不定の繰り返し等には対応できず、柔軟な表現が困難。
通信データ量	△ 電文項目とデータというシンプルな構造のためデータ量が少ない。	× 終了タグが必要であり、電文項目が多いほどデータ量が増大する。	電文項目が電文内に存在せず、データ量は最も少ない。
処理性能	○ JavaScript のオブジェクト記法で、データ解析が容易で処理負荷が低いとされる。	△ データを扱うために構文解析を要し、データ量に応じて処理時間がかかる場合がある。	データ構造がシンプルであり、データ処理が容易。
その他	全銀協「オープン API のあり方に関する検討会報告書」で推奨されているとおり、開発標準として主流。	SWIFT が ISO20022 の標準フォーマットとして採用。一方で、標準ライブラリで非推奨となっている言語がある等、将来性に懸念がある。	－

電文項目については、「電文種目ごとに定義」する方式と「全電文項目共通で定義」する方式が選択肢として挙げられる。電文種目ごとに個別に電文を提示する場合は、電文項目は電文データの意味と一致するため、参加者は電文項目名に応じて容易に電文データを設定できる。一方、全電文項目を共通で定義する場合は、電文項目は電文データの意味とは関係なく共通化されるため、参加者は通信種目ごとに用途を読み替えて電文データを設定する必要がある。なお、全銀システムは全電文種目共通で定義されている。それぞれの電文仕様のイメージを図表 4.17 に示す。

図表 4.17　電文仕様



図表 4.18 のとおり、電文仕様の比較評価の結果、全電文種目で共通フォーマットを採用した場合、現行の全銀システムと親和性が高くなるため、API ゲートウェイ側の実装負担は低減できる可能性があるものの、参加者システムにおける可読性やシステム実装負担を考慮し、本実証実験では、電文項目等ごとに定義することとした。

図表 4.18　電文仕様の比較評価

観点	電文種目等ごとに定義	全電文種目共通で定義
可読性	〇 電文項目と電文データの意味が一致するため、可読性は高い。	× 電文項目と電文データの意味が一致しないため、可読性は低い。
変更容易性	〇 電文項目名に対して 1 つの電文データが意味付けされているため、変更に関して他の電文項目への影響が少ない。	△ 複数の電文項目に跨る電文データが存在するため、任意の電文項目の変更に際して他の電文項目への影響も考慮する必要がある。
システムの実装負担（参加者）	〇 参加者側においてはエンドユーザからの入力情報とのマッピングが容易と想定される。	× 参加者側において用途ごとに電文項目の読み替えが必要。
システムの実装負担（API ゲートウェイ）	△ 現行全銀システムの電文フォーマットへの変換が複雑であり、取扱種目ごとに異なる変換ロジックが必要となる。	〇 全銀システムの電文フォーマットへの変換が容易であり、全通信種目で同じロジックでの実現が可能。

(3) 認証方式

本実証実験では、専用の回線や機器の導入が不要なインターネットを前提に、上位レイヤでの複数の対策として、「IP アドレス認証」、「証明書認証（相互 TLS）」、「API キー認証」を組み合わせた認証方式を採用した。採用案の例として挙げられるネットワークおよび認証方式の概要を図表 4.19 に示す。

図表 4.19 ネットワークおよび認証方式の概要

カテゴリ	主な選択肢	セキュリティに関する概要	負担
ネットワーク	インターネット	不特定多数からの攻撃や盗聴等のリスク。TLS(https)による対策が一般に普及。	一般的なインターネット契約のみ。
	インターネット VPN	インターネット上に構築した仮想閉域網で、特定参加者間でのセキュアな通信を実現。	VPN 接続のための専用装置等が必要。
	IP-VPN／広域イーサネット	ネットワーク層で論理分割することで、特定参加者間でのセキュアな通信を実現。	VPN 接続のための専用装置等に加え、専用のアクセス回線も必要。
	専用線	物理的に分離された接続で、セキュリティ強度は高い。	専用回線に加え、拠点間の距離等に応じた費用が必要（相対的に高額）。
接続先認証	証明書認証（相互 TLS）	経路上の盗聴、改ざん、なりすましへのアプリケーション層での対策として代表的。接続元先での相互認証に加え、通信を暗号化。	導入は容易（アプリケーションへのインポート）。失効・再発行運用等が必要。
	IP アドレス認証	ネットワーク層での接続相手を限定できる。	接続元アドレスを固定する費用や制約。
アプリケーションでの認証	API キー認証	管理者発行のキーを設定したリクエストのみ許容する。単体の認証強度は高くない。	導入は容易（リクエストへの設定）。 保管、変更管理の運用が必要。
	Basic 認証	ID・パスワードの組み合わせ等で認証。単体の認証強度は高くない。	導入は容易（リクエストへの設定）。 保管、変更管理の運用が必要。
	Digest 認証	ID／パスワードを（Nonce 等と合わせて）ハッシュ化して認証。盗聴対策となる。	導入がやや煩雑（Nonce 処理等）。 保管、変更管理の運用が必要。

5 実証実験結果

5.1 機能面（基本系・冗脱管理）

(1) 検証結果

機能面において、論点となる処理方式でのテレ為替業務が実施可能か検証を行った結果、ノックアウトファクターとなる問題は検出されなかった。観点ごとの検証結果を図表 5.1 に示す。

具体的な検証方法としては、検証項目および確認手順を詳細化したうえで、参加者システム側で準備したクライアントアプリケーションから確認手順に沿って API を実行し検証を行った。あわせて API ゲートウェイの処理結果からも、参加者の検証が問題なく行われていることを確認した。

図表 5.1 機能面の検証結果

項目		検証項目	検証項目（詳細）	検証結果※
基本形	仕向観点	参加者システムから上り電文の送信、API ゲートウェイからの処理結果の受信が API 仕様書どおり実現できること	API ゲートウェイで正常に処理される上り電文の送信	○
			API ゲートウェイでエラーとなる上り電文の送信	○
			上り電文送信時に設定するリクエスト ID と電文内容の組み合わせに関する挙動確認	○
	被仕向観点	参加者システムからの下り電文配信依頼（ポーリング）、API ゲートウェイからの下り電文の配信が API 仕様どおり実現できること	振込（当日）電文の受信	○
			エラー通報電文の受信	○
			複数種類の下り電文の同時受信	○
		1 回のポーリングで配信された複数の下り電文を参加者システムで処理（入金等）できること	参加者システムへの振込（当日）電文の連携	○
冗脱管理／対処		ユニークな ID が付与された電文それぞれに対して、API ゲートウェイ／参加者システムにおいて、抜け漏れなく処理できること	上り電文の処理状況の確認	○
			下り電文の受信状況の確認	○
		ID かつ内容が重複した上り電文を受信した場合、API ゲートウェイで二重処	同一の上り電文送信時の挙動確認	○

項目	検証項目	検証項目（詳細）	検証結果※
	理しないこと、また、ID が重複した下り電文を再取得した場合、参加者システムにおいて二重処理しないこと	同一の下り電文受信時の挙動確認	○
	上り電文やポーリングの結果が一定時間以上返却されない場合、参加者システムでタイムアウトを検知し、再送できること、また、再送電文が正常に処理されること	クライアントにおけるタイムアウト検知時の挙動確認	○

※○：ノックアウトファクターは検出されなかった。

5.2　非機能面（処理性能）

(1)　検証結果

論点となる処理方式における性能への影響について検証した結果、ノックアウトファクターとなる問題は検出されなかった。観点ごとの検証結果を図表 5.2 に示す。

なお、本実証実験の実施時点では API ゲートウェイに求められる性能要件が未定であったため、現行の全銀システムにおける加盟銀行の取引量を参考に、API ゲートウェイにおける条件について仮定を立て検証を行った。

図表 5.2 処理性能の検証結果

項目	検証観点	検証結果※
接続先単位および API ゲートウェイ全体のスループット 各 API のレスポンスタイム	・ 現行の加盟銀行の単位時間あたり取引量等を踏まえた負荷モデルを用いて確認する ・ データアクセス競合等によりスケールアウトでは解決できないボトルネックが存在しないか確認する ・ 各 API の処理性能が、処理特性（参照系・更新系等）と大幅に乖離が無いか確認する	○
キャパシティ要件の性能への影響	・ 現行の加盟銀行の 1 日あたり取引量と電文保管要件を踏まえた格納データ量で測定する ・ データ蓄積時も大幅な性能劣化が無いことを確認する	○
ポーリングによる仕向から被仕向への電文到達時間への影響	・ ポーリング 1 回あたりの取得電文数を変化させた場合における処理時間、取得効率、システム負荷への影響を確認する	○

※　○：ノックアウトファクターは検出されなかった。

(2) 試験条件

処理性能の検証における環境スペックを図表 5.3 に示す。

図表 5.3 環境スペック

分類	スペック
API（アプリケーション）	CPU：2vCPU メモリ：1GB
DB（RDB）	CPU： 8 vCPU メモリ：64GB

① 接続先単位および API ゲートウェイ全体のスループット

現行の加盟銀行の RC 最大疎通能力を参考に、API ゲートウェイの接続先数および接続先ごとの疎通量を仮定して負荷モデルを設定した（図表 5.4）。負荷モデルの設定の考え方は以下のとおり。

- API ゲートウェイは RC と同様に東阪両現用と仮定し、片センタにおける RC 最大疎通能力をもとに設定した。
- API ゲートウェイの接続先数は現時点で未定であるが、目安として現行の 4 分の 1 程度を前提とした。また接続先数の分布は、現行の加盟銀行の分布と近似するよう設定した。
- 現行の加盟銀行において最大となる RC 疎通能力（497,600 件）は、API ゲートウェイの要件として必要か現時点で未定であるが、システム能力として最大値での実行が可能か検証する目的で対象とした。

負荷モデルとして設定した疎通量を目標値とし、当該値を下回らないよう負荷試験ツールを用いてリクエストを連続的に送信し、上り電文送信、下り電文配信要求のスループットを測定した。

なお、参考とした現行の加盟銀行ごとの RC 最大疎通能力と銀行数の分布を図表 5.5 に示す。

図表 5.4 本実証実験における負荷モデル

接続先ごとの疎通量（件／時）	該当する接続先数	接続先数の分布
497,600	1	3%
62,200	1	3%
46,700	1	3%
31,100	4	13%
15,600	7	23%
7,700	8	27%
3,900	8	27%
合計	30	—

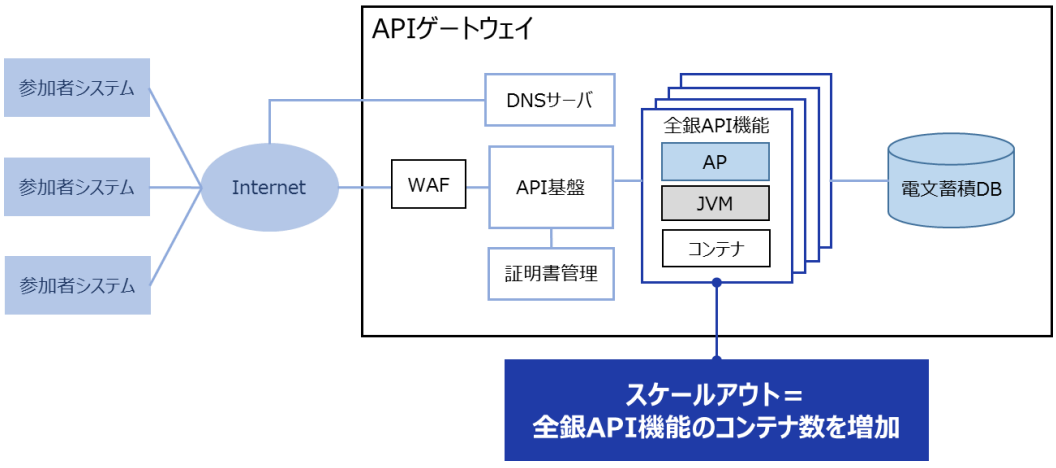
図表 5.5 加盟銀行ごとの RC 最大疎通能力と銀行数の分布（2021 年 7 月時点）

銀行ごとの RC 最大疎通能力（片センタ）（件/時）	該当する銀行数	銀行数の分布
497,600	2	1%
394,800	1	1%
313,000	1	1%
217,700	1	1%
124,500	1	1%
93,300	1	1%
62,200	3	2%
46,700	6	4%
31,100	19	14%
15,600	31	23%
7,700	34	25%
3,900	34	25%
合計	134	—

※RC を複数台保有している先は台数分の合算を計上。

また上記に加えて、スケールアウトによって処理性能の向上が可能か確認するために、全銀 API 機能のコンテナ数を 10～60 の範囲で 10 ずつ増加させ、各条件における上り電文送信のスループットを測定した（図表 5.6）。また、本検証においては、図表 5.7 の環境スペックで検証を行った。

図表 5.6 スケールアウト



図表 5.7 スケールアウトの検証における環境スペック

分類	スペック
API（アプリケーション）	変更なし
DB（RDB）	CPU：32vCPU メモリ：256GB

② 各 API のレスポンスタイム

上り電文送信、下り電文配信要求は、スループットの試験時に同時にレスポンスタイムを測定した。

上り電文照会、下り電文照会は、スループットの試験条件下において複数回実行し、サーバ負荷状況下におけるレスポンスタイムを測定した。また、照会機能の処理特性を考慮し、照会対象として取得される件数を 1、5,000、10,000、15,000、20,000、25,000 件のパターンで変化させた場合のレスポンスタイムの変化を測定した。

③ キャパシティ要件の性能への影響

キャパシティの試験条件を設定するうえで、現行の全銀システムにおける電文保管要件をもとに保存期間を仮定した。現行の電文保管要件は、使用用途に応じて、図表 5.8 のとおり、2 パターンが存在する。

図表 5.8 電文保管要件

使用用途	保管期間
発受信の証として保存する	1 ケ月
テレ為替において電文再取得を行う	2 営業日（前営業日・当営業日）

上記から、1 ケ月分の電文データは、取引実行時にリアルタイムで更新されるデータ領域とは別に管理することが適切と考えられる。一方、2 営業日分の再取得用の電文データは、再取得要求に応じて即時に参照する必要があることから、リアルタイムで更新されるデータ領域に保管する必要がある。

本実証実験においては、現行の全銀システムにおける 1 営業日あたりのデータ収容能力の 4 分の 1 である 750 万件※を初期データとして登録した状態でレスポンスタイムを測定した。

※現行の全銀システムにおける 1 営業日あたりのデータ収容能力：片センター当たり 3000 万件。API ゲートウェイの接続先数は現時点で未定だが、目安として現行の 4 分の 1 を前提とした（「① 接続先単位および API ゲートウェイ全体のスループット」の試験条件と同様）。

④ ポーリングによる仕向から被仕向への電文到達時間への影響

下り電文配信要求について、ポーリング 1 回あたりの取得電文数を 25 件、50 件、100 件、200 件、400 件、800 件のパターンで変化させた場合における処理時間、取得効率（単位時間あたりの取得電文数）、リソース使用量を測定した。

(3) 測定結果

① 上り電文送信・下り電文配信要求の負荷条件下でのスループット・レスポンスタイム

上り電文送信、下り電文配信要求について、図表 5.4 で設定した負荷モデルでの負荷条件下のスループット・レスポンスタイムの測定結果を以下に示す。なお、本検証においては、図表 5.4 で設定した負荷モデルの疎通量（件／時）を TPS に換算し、目標値相当のスループットを満たすか確認を行った。検証の結果、すべての接続先において設定した目標値を満たすスループットで処理が実行できることを確認した（図表 5.9）。また、疎通量ごとのレスポンスタイムを比較した結果、疎通量の異なる接続先においても、レスポンスタイムはほとんど差がないことを確認した（図表 5.10、5.11）。これらのことから、今回仮定した負荷モデル（API ゲートウェイの全体スループット：現行の加盟銀行の RC 最大疎通能力合計の約 4 分の 1 程度）の範囲においては、必要なスループットを実現するうえでのボトルネックはないものと考えられる。

図表 5.9 スループット

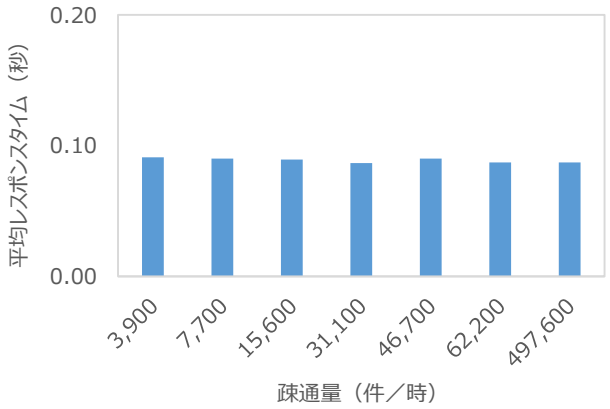
パターン	接続先数	目標値（上り・下り共通）		実績値※1	
		件／時	TPS※2	上り電文送信（TPS）	下り電文配信要求（TPS）
A	1	497,600	139	151	139
B	1	62,200	18	20	18
C	1	46,700	13	14	14
D	4	31,100	9	9	10
E	7	15,600	5	6	6
F	8	7,700	3	4	4
G	8	3,900	2	2	2
合計※3	30	932,900	281	307	300

※1 同パターンの接続先数が複数の場合は平均値を掲載。

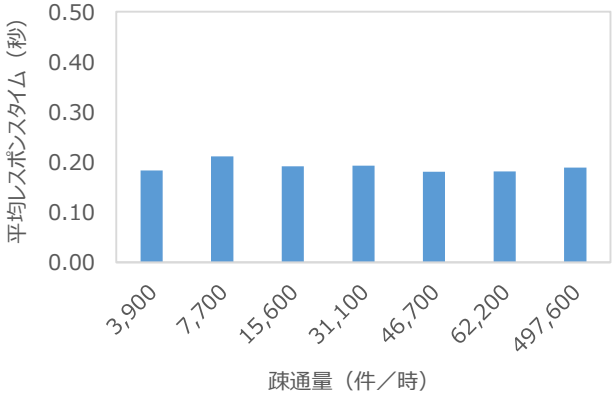
※2 小数点以下端数を切り上げた値。

※3 接続先 30 先の合算値。

図表 5.10 上り電文送信のレスポンスタイム



図表 5.11 下り電文配信要求のレスポンスタイム



また、試験条件で設定した負荷条件下で実施した場合と、無風条件下で実行した場合のレスポンスタイムの比較を図表 5.12 に示す。検証の結果、上り電文送信、下り電文配信要求ともに、負荷条件下においてもレスポンスタイムの劣化はほとんど見られなかった。これらのことから、上り電文送信、下り電文配信要求の処理性能は、今回仮定した負荷条件下（図表 5.4「本実証実験における負荷モデル」）の範囲においては影響を受けないものと考えられる。

図表 5.12 上り電文送信・下り電文配信要求の無風／負荷条件下でのレスポンスタイム比較※1

API	平均レスポンスタイム（秒）		
	無風条件下	負荷条件下※2	差分
上り電文送信	0.09	0.09	±0
下り電文配信要求	0.15	0.18	+0.03

※1 図表 5.4「本実証実験における負荷モデル」における「疎通量：46,700 件／時」の接続先に相当するクライアントの測定値を基準とした。

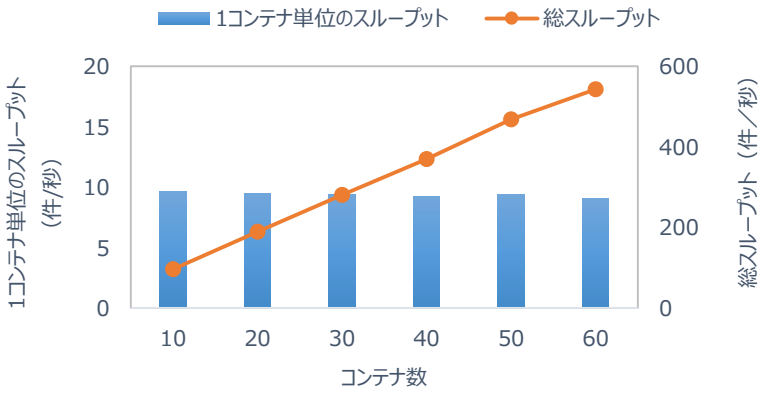
※2 図表 5.4「本実証実験における負荷モデル」の条件に則る。

② スケールアウトによる上り電文送信のスループットの変化

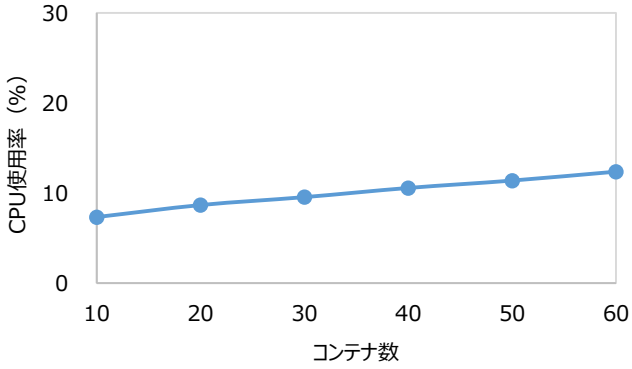
スケールアウトさせた場合（＝全銀 API 機能のコンテナ数を増加させた場合）における上り電文送信のスループットの測定結果を図表 5.13 に示す。なお、本検証においては、スループットを TPS に換算して測定した。検証の結果、今回の検証範囲においては、コンテナ単位でのスループットは大きく劣化することはなく、コンテナ数の増加に応じて線形に総スループットが増加し、最大 500TPS 程度まで向上することを確認した。また、DB リソース使用量の変化については、CPU 使用率、IOPS（1 秒あたりの DB 読み書きレコード数）ともに、スケールアウトに応じて増加する傾向が見られるものの、コンテナ数 10～60 の範囲においては、CPU 使用率はいずれも 10%程度であることを確認した（図表 5.14、5.15）。

これらのことから、今回の検証範囲（コンテナ数 10～60）においては、スケールアウトによってスループットを向上させるうえでのボトルネックはないものと考えられる。

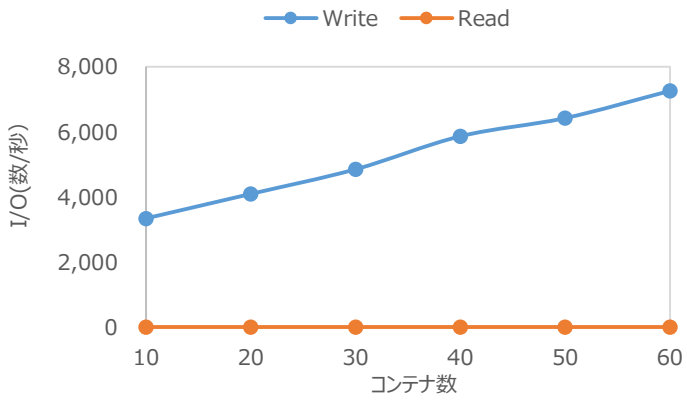
図表 5.13 コンテナ数を増加させた場合の上り電文送信のスループット



図表 5.14 DB CPU 使用率



図表 5.15 IOPS



③ 上り電文照会・下り電文照会のレスポンスタイム

上り電文照会、下り電文照会について、負荷条件下（上り電文送信、下り電文配信要求が、図表 5.4 のとおり並走実行されている状態）において上り電文照会、下り電文照会を実行した場合と、無風条件下で実行した場合のレスポンスタイムの比較を図表 5.16 に示す。検証の結果、上り電文照会、下り電文照会のレスポンスタイムは、負荷条件下においても 0.10～0.30 秒程度の増加であることを確認した。これらのことから、上り電文照会、下り電文照会の処理性能は、今回仮定した負荷条件下（図表 5.4「本実証実験における負荷モデル」）の範囲においては、大幅な影響は受けないものと考えられる。

図表 5.16 上り電文照会・下り電文照会の無風／負荷条件下でのレスポンスタイム比較※1

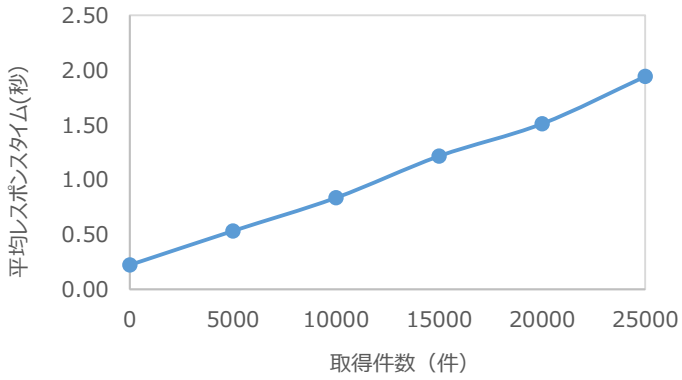
API	平均レスポンスタイム（秒）		
	無風条件下	負荷条件下※2	差分
上り電文照会	1.21	1.32	+0.11
下り電文照会	1.23	1.57	+0.34

※1 照会対象として取得される件数が 15,000 件の場合の測定値を基準とした。

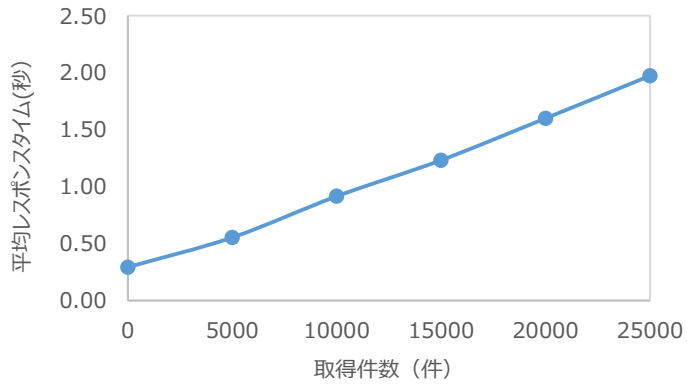
※2 図表 5.4「本実証実験における負荷モデル」の条件に則る。

また、照会対象として取得される件数を 1～25,000 件の間で変化させた場合のレスポンスタイムの測定結果を以下に示す。検証の結果、照会対象の取得件数の増加に応じて線形にレスポンスタイムが増加することを確認した（図表 5.17、5.18）。これらのことから、本実証実験での仕様における上り電文照会、下り電文照会の処理性能は、今回の検証範囲（取得件数 1～25,000 件）においては処理特性に応じた妥当なものであり、問題ないと考えられる。

図表 5.17 上り電文照会のレスポンスタイム



図表 5.18 下り電文照会のレスポンスタイム



④ キャパシティ要件の性能への影響

上り電文送信、下り電文配信要求について、設定した試験条件のとおり、前営業日分の電文情報を 750 万件登録した状態と蓄積データ 0 件の状態において、それぞれ指定した疎通量で実行した場合のレスポンスタイムの比較を図表 5.19 に示す。検証の結果、蓄積データの有無によるレスポンスタイムの差はほとんど発生しないことを確認した。これらのことから、今回仮定したキャパシティ要件（現行の 1 営業日あたりのデータ収容能力の 4 分の 1 である 750 万件の前営業日分の電文情報を最大データ量として登録した状態）においては、上り電文送信、下り電文配信要求の処理性能はほとんど影響を受けないものと考えられる。

図表 5.19 上り電文送信・下り電文配信要求のデータ蓄積時のレスポンスタイム比較※

API	平均レスポンスタイム（秒）		
	データ蓄積なし（0 件）	データ蓄積あり（750 万件）	差分
上り電文送信	0.10	0.09	-0.01
下り電文配信要求	0.17	0.18	+0.01

※「疎通量：46,700 件／時」の接続先に相当するクライアントの測定値を基準とした。

⑤ ポーリングによる仕向から被仕向への電文到達時間への影響

下り電文配信要求について、ポーリング 1 回あたりの取得電文数を変化させた場合の処理時間、単位時間あたりの取得電文数、リソース使用量の測定結果を以下に示す。

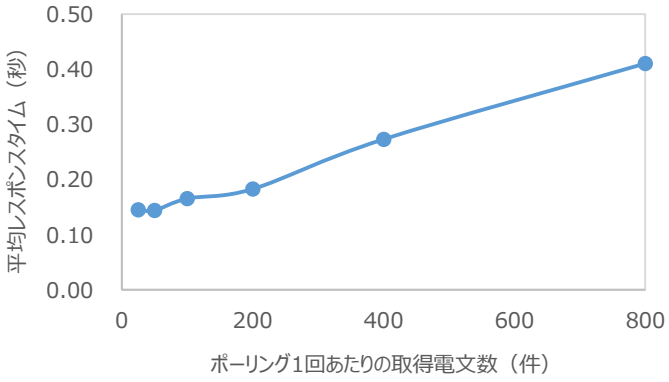
処理時間については、図表 5.20 のとおり、ポーリング 1 回あたりの取得電文数が 50 件以下の場合はほとんど変化が見られず、下り電文配信要求にかかる最小の処理時間は 0.15 秒程度であることを確認した。また、取得電文数が 50 件以上の場合は電文数の増加に応じて処理時間が増加するが、今回の検証範囲で最大となる取得電文数 800 件の場合においても、処理時間は 0.40 秒程度であることを確認した。これらのことから、今回の検証範囲（ポーリング 1 回あたりの取得電文数 25～800 件）においては、ポーリングによる電文到達時間への影響は大きな問題にはならない範囲と考えられる。

また、ポーリング 1 回あたりの取得電文数と平均レスポンスタイムから、1 秒あたりの取得電文数を算出した結果を図表 5.21 に示す。今回の検証範囲においては、ポーリング 1 回あたりの取得電文数が多いほど、時間あたりの取得電文数（取得効率）が高くなり、最大で 2,500 件／秒程度となることを確認した。これらのことから、今回の検証範囲（ポーリング 1 回あたりの取得電文数 25～800 件）においては、ポーリング 1 回あたりの取得電文数を増加させることで取得効率を向上させることによるボトルネックはないものと考えられる。

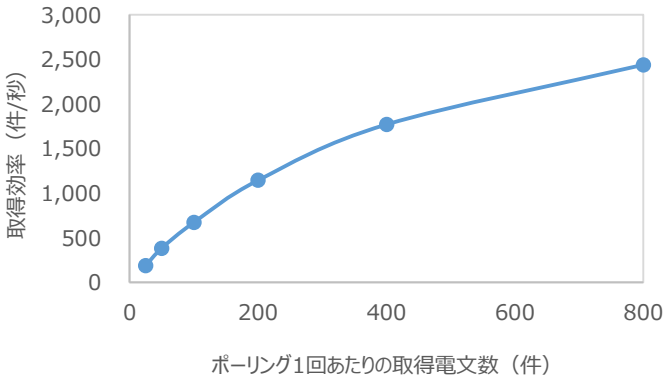
なお、DB リソース使用量の変化については、IOPS（1 秒あたりの DB 読み書きレコード数）はポーリング 1 回あたりの取得電文数の増加に応じて増加するものの、CPU 使用率は大きな変化はないことを確認した（図表 5.22、5.23）。これらのことから、今回の検証範囲（ポーリング 1 回あたりの取得電文数 25～800 件）においては、ポーリングによるリソース負荷への影響は小さいものと考えられる。

以上のことから、今回の検証範囲（ポーリング 1 回あたりの取得電文数 25～800 件）においては、下り電文配信方式は技術的にポーリングで実現可能と考えられる。

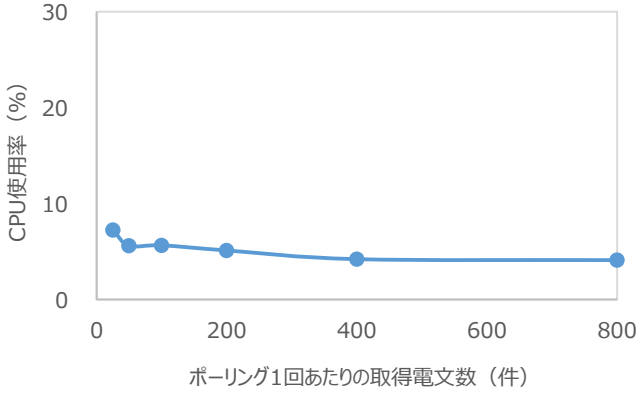
図表 5.20 ポーリング 1 回あたりの処理時間



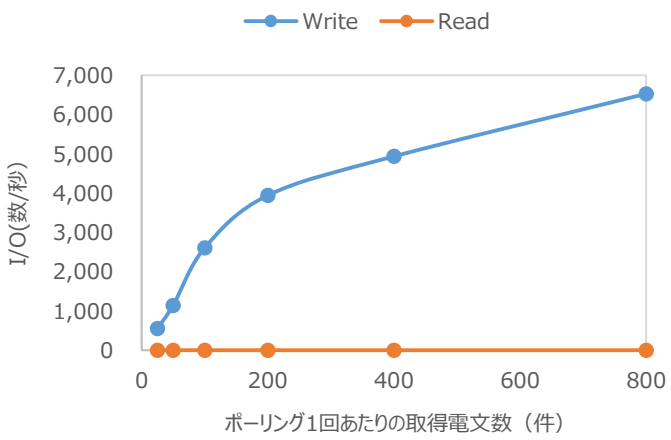
図表 5.21 取得効率（1 秒あたりの取得電文数）



図表 5.22 DB CPU 使用率



図表 5.23 IOPS


Write
Read
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
I/O(数/秒)
0
200
400
600
800
ポーリング1回あたりの取得電文数（件）

5.3　その他

(1)　参加者システムで確認された測定結果

参考情報として参加者システム側で確認された測定結果を以下に示す。

① パターン 1

a 実施内容

参加者システムのクライアントアプリケーションから API をコールし、以下の条件で依頼～応答のレスポンス時間を計測した。

- 上り電文：「No.1　上り電文送信」の API を 10 分間コールし、依頼～応答までのレスポンス時間を計測した。負荷試験中に「No.2　上り電文照会」の API を複数回実行した。
- 下り電文：「No.3　下り電文配信要求」の API を 5 分間コールし、依頼～応答までのレスポンス時間を計測した。負荷試験中に、「No.4　下り電文照会」の API を複数回実行した。

※多重化せず、シーケンシャルに API 実行した。また、上り電文送信を除く 3API については、数秒のインターバルを設けてコールした。

b 測定結果

API ごとの TPS、レスポンスタイムの測定結果を図表 5.24 に示す。

図表 5.24 参加者システム側での測定結果（パターン 1）

	API コール数	TPS	レスポンスタイム（秒）		
			平均	最大	最小
上り電文（10 分間）					
No.1 上り電文送信	4658 回	7.8	0.13	1.12	0.07
No.2 上り電文照会	111 回	(0.2)	0.44	1.36	0.21
下り電文（5 分間）					
No.3　下り電文配信要求	130 回	(0.4)	0.32	0.93	0.18
No.4　下り電文照会	45 回	(0.2)	1.74	4.65	1.34

※No.2～4 は数秒のインターバルを設けてコールしているため、TPS は参考情報。

c 考察

上記測定結果について、API ゲートウェイ内のレスポンスタイムを調査した結果は図表 5.25 のとおりであった。差分はネットワーク伝送時間と想定される。

図表 5.25 API ゲートウェイ内の処理時間（パターン１）

	レスポンスタイム（秒）		
	平均	最大	最小
上り電文（10 分間）			
No.1 上り電文送信	0.10	1.10	0.05
No.2 上り電文照会	0.29	0.67	0.13
下り電文（5 分間）			
No.3 下り電文配信要求	0.22	0.45	0.13
No.4 下り電文照会	1.39	3.30	1.00

② パターン 2

a 実施内容

上り電文送信を一定時間連続実行した場合のレスポンスタイムを測定した。

※使用した VPN は AWS、ドイツ・フランクフルトのデータセンター。

b 測定結果

レスポンスタイムは最小 2 秒、最大 4.5 秒、平均 2.5 秒であった。

c 考察

上記測定結果について、API ゲートウェイ内の処理時間を調査した結果、最小 0.06 秒、最大 0.50 秒、平均 0.13 秒であった。差分はネットワーク伝送時間と想定される。

(2) 参加者からの意見

検証観点以外の検討事項について、参加者から寄せられた意見を以下に示す。

① 照会機能の改善

照会機能（上り電文照会、下り電文照会）について、本実証実験では、全件照会、もしくは電文の処理状況を条件にした絞り込みによる照会を機能として実装したが、リクエスト ID を条件にした単独電文の照会機能の要望が寄せられた。

② 業務エラーのレスポンス内容改善

業務エラーのレスポンス（電文形式誤り）について、具体的に電文のどの部分が不正か示していただくと修正しやすいとの意見が寄せられた。

③ 認証方式の検討

認証方式について、IP アドレス制御は、運用面の負担が考えられること、また、証明書認証（相互 TLS）および API キー認証によって、接続先を制限できることから、必須要件ではないとの意見が寄せられた。IP アドレスはクラウド環境において流動的に変わることが想定されるため、変更の都度 API ゲートウェイでの設定手続きが必要になるとの懸念が挙げられた。

④ 試験環境の提供

事前に API ゲートウェイと接続し、API インタフェースや業務観点の確認ができる試験環境が必要との意見が寄せられた。

⑤ 双方向 API の採用

本実証実験においては、参加者の API 構築負担を考慮し、ポーリングを採用したが、双方向 API でリアルタイムに下り電文を配信してほしいとの意見が寄せられた。

(3) NTT データからの準備・検証期間における気づき

検証観点以外の検討事項について、本検証実験の準備・検証を通じて、NTT データから挙げられた気づき事項を以下に示す。

① インターネット接続に関するセキュリティ対策

本実証実験では、インターネット接続に関して、図表 5.26 のとおりセキュリティ対策を実施した。なお、実証実験の期間中、API ゲートウェイに対するインターネットからの攻撃は検知されなかった。

図表 5.26 本実証実験でのインターネット接続に関するセキュリティ対策

対策	本実証実験での結果
DDoS 保護	クラウドサービスが提供する DDoS 対策を使用。攻撃検知なし。
FW	参加者から事前に提示された IP アドレスのみ許可するルールを設定。攻撃検知なし。なお、登録外の IP アドレスからのアクセスを検知したが、参加者の IP アドレス指定誤りであることを確認済み。
WAF	アプリケーションの脆弱性を狙った攻撃を防御する対策を使用。攻撃検知なし。
マルウェア対策	マルウェア感染や不正アクセスの検知なし。

なお、今回利用したクラウドサービスが発表している最新の脅威動向では、TCP SYN フラッドや UDP リフレクション攻撃といった、インフラストラクチャ層を狙った DDoS 攻撃が年々増加傾向にあると発表しており、実証実験期間中に利用中のクラウドサービスが受けた攻撃で最も多かったのは TCP SYN フラッドであった。こうした DDoS 攻撃に対しては、センタ設置型の対策装置では防ぎきれないことがあるため、本番

環境に向けて、回線事業者や SaaS 事業者が提供する DDoS 対策サービスの利用等を継続検討することが推奨される。その他、近年よく見られる脅威である DNS サーバに対する攻撃（DNS ポイズニング等）に係る対策についても検討することが推奨される。具体的には、DNS キャッシュサーバに対して攻撃を受けにくくする対策（例：DNSSEC、ソースポートランダマイゼーション、DNS クライアントの限定等）や、攻撃を受けて虚偽のサイトに誘導された場合にエラーを検知できるよう、証明書認証（相互 TLS）等の対策を実施することが考えられる。

② 認証情報の流出対策

本実証実験中に申請外の IP アドレスから正規の API キー／証明書を用いたアクセスがあり、参加者に状況確認を行った。本件については、正規の参加者からのアクセスであることが確認できたため、IP アドレスの追加登録する対処を行ったが、API ゲートウェイからだけでは認証情報が流出したか参加者の設定誤りかが判断できない。そのため、本番環境に向けては、参加者と API ゲートウェイ双方で不正取引のリスクに備えた追加的な検討が推奨される。例えば、情報漏洩対策やなりすまし対策、参加者側でアクセス状況を確認する運用等が考えられる。

③ クラウドサービスに依存する仕様の管理方法

本実証実験の API 仕様書で提示した仕様の中には、クラウドサービス仕様に依存する内容が含まれる（TLS バージョン、Cipher Suites 等）。クラウドサービスに依存する仕様は、参加者への通知後にサービスの仕様変更に伴い変更となる可能性があることから、本番環境においてもクラウドサービスを利用する場合は、仕様の開示方法や管理方法について検討することが推奨される。例えば、クラウドサービスベンダーとサポートを組み、密に連携を図ることで、クラウドサービスの仕様変更に係る情報を事前に入手することや、仕様変更に伴う事前動作検証の実施の運用等が考えられる。

④ 回線サービスの障害・緊急メンテナンスへの対策

本実証実験の準備期間において、NTT データの環境構築で利用した回線サービスの障害・緊急メンテナンスによる回線断が発生した。本番環境に向けては、業務継続できるよう経路冗長化等の対策について検討することが推奨される。

6 考察・今後の進め方

6.1 実証実験の結果および検出された課題の考察

本実証実験の結果、採用した処理方式（ステートレス方式／ポーリング方式／ID 方式）でのテレ為替業務（テレ為替電文の送受信）について、基本的な機能性、冗脱管理、処理性能に関してノックアウトファクターとなる問題は検出されなかった。一方、本実証実験を通じて本番環境に向けた検討課題が複数検出されたため、今後の対応の方向性を以下に示す。

(1) 利用者要望を考慮した実装機能の検討

本実証実験を通じて参加者から寄せられた要望を踏まえて、以下のような実装機能の改善を検討する。

① 照会機能の改善

照会機能（上り電文照会、下り電文照会）について、参加者の運用を考慮した単独電文の照会機能や、取得対象の絞り込み条件やソート順の追加が考えられる。参加者からの要望や広く一般的に提供される機能を参考に、標準的なつくりの実現を検討する。

② 業務エラー時のレスポンス情報の改善

参加者の要望等を踏まえ、業務エラー時のレスポンス情報（エラー事象詳細の表示等）を検討する。

③ 双方向 API の採用

参加者の要望等を踏まえ、参加者の運用リアルタイムで下り電文配信をするために、双方向 API の採用が可能か運用面を含めて検討する必要がある。実現方法としては、標準化・軽量の観点から、双方向 API の場合においても、本実証実験で採用した通信方式（ステートレス）および冗脱管理方式（ID 方式）を採用し、上り電文送信と同等の機能を参加者システムに構築することで実現が可能と考えられる。

(2) 試験環境の検討

参加者が事前に API インタフェースや業務機能を検証するために必要な試験環境（サンドボックス環境等）の提供について検討する。

(3) セキュリティ対策の検討

本実証実験で採用した認証方式およびセキュリティ対策（不正アクセス防止、マルウェア対策、DDoS 保護）を踏まえて、本番環境に必要とされるセキュリティ要件に加えて、導入コストや参加者の運用負担等から総合的に採用方式を検討する。

6.2 今後の進め方・スケジュール

API ゲートウェイの検討に係るスケジュールを図表 6.1 に示す。本実証実験の結果と今年度末にかけて並行で行われる「必要機能等検討」の結果を踏まえて、2022 年度上期に「構築方針策定」を行う。構築方針の策定にあたっては、資金移動業者および加盟銀行における API ゲートウェイを通じた全銀システムへの接続見通し（利用開始時期を含む）を踏まえ、構築タイミングや費用負担ルール等を整理する。

図表 6.1 API ゲートウェイの検討スケジュール