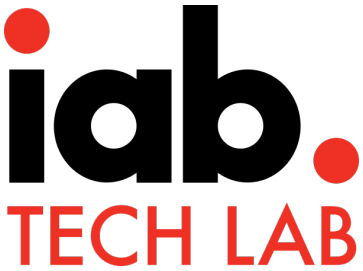

Agentic RTB Framework

OpenRTB およびデジタル広告におけるエージェント駆動型コンテナの使用に関する仕様

Version 1.1

Released November 12, 2025

ご意見やご質問は support@iabtechlab.com までメールでお寄せください。このドキュメントは https://iabtechlab.com/standards/artf/ でオンライン公開されています。

Translation: Advertisers and Publishers Transparency Initiative (APTI)
Original Author: IAB Technology Laboratory (IAB Tech Lab)

About this document（このドキュメントについて）

Agentic RTB Framework（エージェンティック RTB フレームワーク）仕様は、ホストプラットフォーム内で動作し、オーケストレーションプラットフォームが共有目標を達成するために直接呼び出すことができるエージェントサービスを実装するための基盤を定義します。このモデルは、ホストのインフラストラクチャにデプロイされたコンテナを活用して、ビッドストリーム処理の重要な側面を、最小限のコスト、レイテンシ、および運用への影響で、一貫した方法でサービスエージェントに委譲することを可能にします。このフレームワークは、コンテナのランタイム動作の標準要件を確立し、信頼性が高く、保護され、プライベートなビッドストリームのミューテーション（変更）を可能にする API を定義することによってこれを実現します。

このアプローチにより、サービスプロバイダーは自社の提供物を一度パッケージ化すれば、あらゆる標準準拠のプラットフォームにデプロイできます。つまり、運用の懸念やスケーリングをホストプラットフォームにオフロードしながら、独自の価値提案に集中できます。また、ホストプラットフォームはデータと SLA（サービス品質保証）の制御を維持するため、情報漏洩、不正流用、レイテンシの懸念なしに、サービスエージェントに対してデータへのより大きなアクセス権とより多くの対話機会を提供できるため、イノベーションの新たな可能性も生まれます。
Agentic RTB Framework は、最小限の統合オーバーヘッドで新機能を「ドロップイン」し、ターゲットユースケースに合わせて特別に構成された入札処理パイプラインを構成できるホストプラットフォームに、大きな価値を提供します。また、この標準により、プラットフォームは、運用コストと要件の制御を維持し、統合にかかる多大なオーバーヘッドやコストを発生させることなく、必要に応じてコンポーネントを追加、更新、削除するだけで、変化する市場の需要に迅速に適応できます。
このアプローチは本質的にエージェンティック（主体的）なものですが、ここでの主な焦点は体系的なエージェンティック統合（サービス間統合）にあります。ただし、統合技術が成熟するにつれて、自律的エージェンティック機能（モデル対サービス）もこの仕様の一部として想定されています。
多くのユースケースがあります。たとえば、エージェントによって提供される ID 解決、取引（Deal）やセグメンテーションのアクティベーション、インプレッション前の不正検出などです。この仕様の目標の一部は、エージェンティックフレームワークを介して新しいユースケースをビッドストリームに統合できるようにすることであるため、このユースケースのリストはここでは完全に列挙されていません。
この仕様は、これらのエージェントをデプロイおよび運用するための一般的なフレームワークとベストプラクティスを提供することを目的としています。事前に定義された一連のユースケースに限定されるものではありません。各ユースケースは、仕様を使用した「意図（Intent）」を介してサポート

されることを意図しています。
さらに、この仕様では、AI エージェントを使用してコンテナを管理できる標準インターフェースについて説明しています。これにより、エージェンティックシステムによって駆動されるサブミリ秒のリアルタイムビディング操作が可能になります。
このドキュメントは主に技術者、特にホストされたコンテナと AI エージェントで解決できる製品や機能を実装したいエンジニアやプロダクトマネージャーを対象としています。読者にとっての重要なポイントは次のとおりです。

- 特定のユースケースにコンテナと AI エージェントを使用する理由を理解する
- コンテナのデジタルフォーマット、要件、機能など、標準的なコンテナデプロイメントに含まれる内容を学ぶ
- コンテナの機能とサービス定義を宣言する方法を学ぶ
- ユースケースとワークフローの例を理解する
- 業界全体での採用を促進するためのベストプラクティスの推奨事項

このドキュメントは、Programmatic Supply Chain Working Group のサブグループである IAB Tech Lab Container Project Task Force によって開発されました。

License (ライセンス)

Significant Contributors (主な貢献者)

Joshua Prismon, Index Exchange; Joe Wilson, Chalice; Arpad Miklos, The Trade Desk; Brian May, Individual; Roni Gordon, Index Exchange; Ran Li, Index Exchange; Ben White, OpenX

IAB Tech Lab Leads (IAB Tech Lab リード)

Miguel Morales, Director Addressability & Privacy Enhancing Technologies (PETs) Shailley Singh, EVP Product & COO

About IAB Tech Lab（IAB Tech Lab について）

IAB Technology Laboratory は、グローバルな業界技術標準とソリューションを作成し、企業の導入を支援する非営利の研究開発コンソーシアムです。Tech Lab の目標は、デジタル広告およびマーケ

ティングのサプライチェーンに関連する摩擦を減らしながら、業界の安全な成長に貢献することです。

IAB Tech Lab は、技術標準の開発を主導し、IAB 標準の迅速かつ費用対効果の高い実装を支援するコードライブラリを作成・維持し、企業が自社の技術ソリューションと IAB 標準との互換性を評価するためのテストプラットフォームを確立しています。IAB 標準は 18 年間にわたり、デジタル広告サプライチェーンにおける相互運用性と収益性の高い成長の基盤となってきました。IAB Technology Lab の詳細については、https://iabtechlab.com をご覧ください。

Disclaimer (免責事項)

THE STANDARDS, THE SPECIFICATIONS, THE MEASUREMENT GUIDELINES, AND ANY OTHER MATERIALS OR SERVICES PROVIDED TO OR USED BY YOU HEREUNDER (THE "PRODUCTS AND SERVICES") ARE PROVIDED "AS IS" AND "AS AVAILABLE," AND IAB TECHNOLOGY LABORATORY, INC. ("TECH LAB") MAKES NO WARRANTY WITH RESPECT TO THE SAME AND HEREBY DISCLAIMS ANY AND ALL EXPRESS, IMPLIED, OR STATUTORY WARRANTIES, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AVAILABILITY, ERROR-FREE OR UNINTERRUPTED OPERATION, AND ANY WARRANTIES ARISING FROM A COURSE OF DEALING, COURSE OF PERFORMANCE, OR USAGE OF TRADE. TO THE EXTENT THAT TECH LAB MAY NOT AS A MATTER OF APPLICABLE LAW DISCLAIM ANY IMPLIED WARRANTY, THE SCOPE AND DURATION OF SUCH WARRANTY WILL BE THE MINIMUM PERMITTED UNDER SUCH LAW. THE PRODUCTS AND SERVICES DO NOT CONSTITUTE BUSINESS OR LEGAL ADVICE. TECH LAB DOES NOT WARRANT THAT THE PRODUCTS AND SERVICES PROVIDED TO OR USED BY YOU HEREUNDER SHALL CAUSE YOU AND/OR YOUR PRODUCTS OR SERVICES TO BE IN COMPLIANCE WITH ANY APPLICABLE LAWS, REGULATIONS, OR SELF-REGULATORY FRAMEWORKS, AND YOU ARE SOLELY RESPONSIBLE FOR COMPLIANCE WITH THE SAME, INCLUDING, BUT NOT LIMITED TO, DATA PROTECTION LAWS, SUCH AS THE PERSONAL INFORMATION PROTECTION AND ELECTRONIC DOCUMENTS ACT (CANADA), THE DATA PROTECTION DIRECTIVE (EU), THE E-PRIVACY DIRECTIVE (EU), THE GENERAL DATA PROTECTION REGULATION (EU), AND THE E-PRIVACY REGULATION (EU) AS AND WHEN THEY BECOME EFFECTIVE.

Glossary（用語集）

Term	Description
Agent	ビッドストリーム内で、特定の自律的な機能（不正検出、取引のキュレーションなど）を実行する、コンテナにカプセル化されたソフトウェアサービス。エージェントは、明示的な意図（intent）を持って、オーケストレーターが定義した制約内で動作するように設計されています。
Agent Manifest	エージェントの機能、リソース要件、意図、依存関係を定義する、コンテナイメージのメタデータとして埋め込まれた JSON 構造。オーケストレーターがコンテナを安全にデプロイおよび管理するために使用されます。
Agent Orchestrator	ビッドストリームに参加するエージェントコンテナのデプロイ、実行、調整を管理するホストプラットフォーム内の制御レイヤー。
Agent to Agent (A2A)	2 つの自律エージェント間の直接的な通信または調整。中央のオーケストレーターによる仲介を必要とせずに、データや決定を交換できるようにします。
Agentic System	定義された制約と相互運用性を維持しながら、オークション結果の最適化などの共有目標に向けて自律エージェントが独立して決定を下す分散アーキテクチャ。
Autonomous Agentic	直接的なオーケストレーションなしに、学習モデルやコンテキストの目標に基づいて動作し、独立した AI 駆動の決定と行動を行うシステムまたはエージェントを指します。
Bidstream	プログラマティック広告エンティティ（SSP、DSP、エクスチェンジ）間で交換されるビッドリクエストとレスポンスの流れ。ARTF では、エー

	ジェントはオーケストレーターの監督下でビッドストリームデータを変更できます。
Bidstream Mutation	OpenRTB パッチを介して表現される、ビッドストリーム内のデータに対する提案された変更（追加、削除、または更新）。各ミューテーションはアトミックであり、特定の意図（intent）に関連付けられています。
Container	エージェントのコード、依存関係、構成を単一のイメージにパッケージ化する、標準化された軽量でポータブルな実行環境。コンテナは OCI ランタイム標準（Docker、Kubernetes など）に準拠しています。
Intent	エージェントがミューテーションを提案する理由（例：「adjustBid」、「activateSegments」、「expireDeals」）を記述した宣言的なステートメント（意図）。意図は、ミューテーションを受け入れるか拒否するかについてのオーケストレーターの意思決定を導きます。
Manifest (Container Manifest)	コンテナのリソース、意図、依存関係、ヘルスチェックを定義するメタデータファイル（通常は container.json）。オーケストレーターが一貫してコンテナをデプロイできるようにします。
MCP (Model Context Protocol)	JSON-RPC を使用した構造化されたモデル対エージェント通信のためのプロトコル。gRPC を補完し、自律的（AI 駆動）なエージェンティックインタラクションをサポートし、モデルがサービスを直接オーケストレーションできるようにします。
Mutation	ビッドリクエストまたはレスポンスに対して提案される単一のアトミックな変更。intent、op（追加/置換/削除）、path（OpenRTB ペイロード内の意味的な参照）などのフィールドが含まれます。

Orchestrator (Orchestrating Entity)	エージェントコンテナの実行を管理し、アクセスを制御し、ミューテーションを検証し、提案された変更を適用するかどうかを決定するホストプラットフォーム（例：SSP、DSP、エクスチェンジ）。
Patch	OpenRTB リクエストまたはレスポンスを変更するためにエージェントによって提案されたミューテーションの構造化されたセット。各パッチはアトミックで追跡可能であり、オーケストレーターの承認が必要です。
Sidecar	プライマリアプリケーションコンテナと一緒に実行され、補助機能（監視、テレメトリ、仲介など）を提供するセカンダリコンテナ。
Structural Agentic	直接的な AI モデル制御ではなく、構造化されたサービス間の相互作用（サービス対サービスオーケストレーション）として相互作用が行われるエージェンティックシステム。
Telemetry	コンテナによって出力される監視データ（メトリクス、トレース、ログ）。オーケストレーションされたシステム全体でのパフォーマンス、セキュリティ、決定の影響を追跡します。
User Cohorts /	Audience Segments オーディエンスセグメンテーションのユースケースの一部としてエージェントがアクティブ化または変更できる、共有された特性または行動を持つユーザーのグループ。

目次

Introduction（はじめに）

What is an agentic system? (エージェンティックシステムとは？)

デジタル広告エコシステムは、さまざまなビジネス活動にわたる多者間参加を可能にする分散システム上に構築されています。この分散システムには共通の目標がありますが、その目標を達成するために他のパートナーによって呼び出される個々のシステムがあります。これらの分散システムのそれぞれは、独自のマニフェストを達成するために独自の目標を達成するようにも機能し、グローバルな目標（つまり、特定のオークション）と独自の目標およびマニフェストの両方を達成するために独自の決定を下します。

構造的エージェンティックシステムは、自律コンポーネントが、定義された制約内でドメイン固有の目標を達成するために、明示的なマニフェスト、意思決定権限、および構造化された委譲パターンを持って動作する分散アーキテクチャです。サプライサイドプラットフォーム（SSP）とデマンドサイドプラットフォーム（DSP）は、OpenRTB などの標準化されたプロトコルを介して他のエージェントに専門的なタスクを委譲することにより、リアルタイムオークションをオーケストレーションする独立したエージェントとして機能します。各エージェントは内部状態を維持し、個別の目的（例：パブリッシャーの収益最大化と広告主の ROI 最大化）に向けて最適化し、厳格なレイテンシ予算内で自律的な決定を下します。一方、システム全体の動作は、集中管理ではなく、これらの専門化されたコンポーネントの相互作用から生まれます。このアーキテクチャパターンは、構成可能性（コンポーザビリティ）、障害分離、独立した最適化、およびコアオークションロジックを変更せずに新機能を統合する機能を可能にし、単一のコンポーネントがグローバルな知識と意思決定権限を持つモノリシックシステムとは根本的に異なります。
この標準では、コンテナと OpenRTB パッチという 2 つの新しいコンストラクトを導入することで、既存のオーケストレーションに新しいエージェントを追加する機能を導入しています。このドキュメントの焦点は RTB プロセスのエージェンティック拡張性を可能にすることですが、コンテナと OpenRTB パッチはどちらも汎用的なエージェンティックアプローチであり、非リアルタイムビディングのユースケースにも使用できます。

Requirements (要件)

エージェンティック拡張性が OpenRTB エコシステムで広く採用されるためには、実装者は基本的な原則のセットに従う必要があります。これらの原則は次のとおりです。

1. **エージェントはコアビッドストリームに参加できなければならない。** ビッドストリームには、

コンテナとの統合ポイントが多数ある場合があります。この標準は、リアルタイムでビッドストリームを介して取引を行っているエンティティに焦点を当てており、エッジでのコンテナ化されたサービスのための汎用的な標準ではありません。

2. **エージェントは特定の目標を達成しなければならない。** 各エージェントは、それ自体の特定の意図（intent）と、ビッドストリームを移動するオークションに対して行いたい変更を宣言しなければなりません。その後、オーケストレーションエンティティは、多者間参加を促進するために変更を受け入れるか受け入れないかを選択できます。
3. **エージェントは、標準的なエンタープライズインフラストラクチャで構成可能かつデプロイ可能でなければならない。** エージェントはコンテナとして構造化され、標準の OCI ランタイムおよびイメージ仕様（つまり、Docker コンテナ）に準拠しなければなりません。これにより、Kubernetes、Docker Compose / Swarm などの分散クラスターオーケストレーションシステムや、Amazon の Elastic Containers サービスなどのクラウドベースのシステムを介して管理できるようになります。
4. **エージェントはパフォーマンスが高く効率的でなければならない。** エージェントは、gRPC や場合によっては MCP などの高性能プロトコルを介して通信しなければなりません。エージェントは効率的な言語で記述し、効率的なエコシステム（たとえば、Rust、Go、Java）または高度に最適化されたインタプリタ言語を活用するべきです。エージェントは、不要なリソースを消費しないべきです。オーケストレーションエンティティは、コンテナプロバイダーに期待される応答時間に関するガイダンスを提供しなければなりません。
5. **エージェントは、最小特権と最小データのポリシーに従わなければならない。エージェントは外界への無制限のアクセスを持たず**、オーケストレーションエンティティが提供する適切なサービスを活用しなければなりません。オーケストレーションエンティティは、コンテナがこの要件に準拠することを保証するために適切な措置を講じるべきです。

What is a Container? (コンテナとは？)

コンテナは、元々 Docker によって普及した標準技術であり、アプリケーションとそのすべての依存関係（コード、ランタイム、システムツール、ライブラリ、設定）を単一のパッケージにカプセル化する、イメージベースの、バージョン管理された、軽量でポータブルな実行環境を提供します。このアプローチにより、ローカル、プライベートデータセンター、またはクラウドのいずれで実行されているかに関係なく、コンテナ化されたアプリケーションがさまざまなコンピューティング環境間で一貫して動作することが保証されます。

コンテナは、各コンテナがホスト OS カーネルを共有しながら独自のランタイムイメージで分離して

実行される、オペレーティングシステムレベルの仮想化の形式を提供します。これらは Open Container Initiative (OCI) 標準を使用して構築されており、イメージ形式とランタイム仕様が定義されているため、Kubernetes、Docker Compose、または Amazon Elastic Container Service (ECS) などのクラウドベースのサービスなど、さまざまなオーケストレーションプラットフォーム間で相互運用可能です。

OpenRTB とプログラマティック広告のコンテキストでは、コンテナは、リアルタイムでビッドストリームと対話する必要があるビジネスロジックの実行ユニットとして機能します。これらの実行ユニットは、入札変更、オーディエンスのアクティベーション、取引（Deal）のキュレーション、その他のビッドストリームプロセスのロジックをホストできると同時に、コンテナまたはオーケストレーターが各パートナーに特注のコアプラットフォームインフラストラクチャを提供する必要なく、参加者に柔軟性、モジュール性、および簡素化されたデプロイメントを提供します。

なぜコンテナなのか？

コンテナは、ポータブル（多くの環境で実行可能）、コンポーザブル（他のコンテナやシステムと組み合わせて価値を提供可能）、動的スケーラブル（需要に応じて容量を増減可能）、および保護されたモード（入口と出口を管理し、IP と機密データを保護して、多様なプログラマティック環境全体で意思決定ロジックを実行可能）を可能にします。コンテナの標準化されたパッケージングにより、パートナーは一度デプロイすれば、再ツール化することなく複数のオーケストレーションエンティティやクラウドインフラストラクチャ全体で運用できます。一方、コンポーザビリティ（構成可能性）により、バイヤー、セラー、ID プロバイダー、測定ベンダーからのロジックを、まとまりのあるリアルタイムオークションワークフローに統合できます。

重要なのは、コンテナがビジネスロジックを自己完結型のイメージにカプセル化し、独自の IP を分離しつつ、独立したスケーリングを必要とせずにビッドストリームへの参加を可能にすることです。

コンテナエコシステムへの統合

パートナー企業からのコンテナをオーケストレーターのインフラストラクチャで実行するには、両者が特定の要件と制限に対処する必要があります。

- エージェントプロバイダーは、アプリケーションが非 root ユーザーとして適切に実行されることを保証しなければなりません。
- オーケストレーターは、コンテナを root ユーザーとして実行してはなりません。
- コンテナは、Kubernetes 互換のヘルスおよびレディネスプローブ（HTTP エンドポイント、TCP チェック、または exec コマンド）を実装しなければなりません。

- コンテナは最小特権の原則に従い、不要な機能をすべて削除しなければなりません。
- コンテナは、グレースフルシャットダウンを処理し、セキュリティコンテキストを尊重するように構築されなければならず、特権アクセスやホストネットワーク/PID 名前空間を必要としてはなりません。
- コンテナは外部ネットワークアクセスなしで実行されなければなりません。オーケストレーションエンティティとのサービス通信を除き、すべてのネットワークの ingress（入来）および egress（送出）は禁止されています。エージェントプロバイダーとオーケストレーションエンティティは、オーケストレーターの環境でのコンテナの効率的な運用を促進するために、特定のネットワークアクセスポリシーについて交渉することができます。

オーケストレーション側では、オーケストレーターは、コンテナに必要な最小限のリソース権限のみを付与する適切な RBAC ポリシーを実装する必要があります。ネットワークポリシーは、ingress トラフィックと egress トラフィックの両方を制御し、コンテナが通信できるサービスとアクセス可能なポートを明示的に定義するように構成する必要があります。ネットワークトラフィック制御の必要なセキュリティレベルに応じて、オーケストレーターは、コンテナのデータのインポートとエクスポートを可能にする外部データ同期機能を備えたボリュームマッピングなどの代替手段をエージェントプロバイダーに許可したい場合があります。シークレット管理の場合、オーケストレーターは、Kubernetes Secrets または同様の方法（マウントされたシークレットファイルシステムや安全な環境変数など）を使用して、必要な認証情報、API キー、または証明書を安全に注入し、それらが適切にスコープ設定され、ローテーションされることを保証しなければなりません。さらに、オーケストレーターは、リソースのクォータと制限を実装し、Pod Security Policies または Pod Security Standards または同様の方法を使用してセキュリティ制約を強制し、インフラストラクチャ内でのコンテナの動作を追跡するための監視とログ記録をサポートすることが期待されています。

なぜ OpenRTB パッチなのか？

OpenRTB 自体は、多者間の構造的エージェンティックシステムです。多くの異なる当事者間の複雑な相互作用は、リアルタイムビディングプロセスのパラメーターを記述する OpenRTB および AdCOM ドメインモデルに組み込まれています。完全な OpenRTB ペイロードを新しいエージェントに渡し、エージェントが完全な更新された OpenRTB ペイロードを返すことによって、既存のシステムに新しいエージェントを導入することには、多くの欠点があります。

- すべてのエンティティに完全な OpenRTB ペイロードを渡す非効率性は、かなりのシリアル化およびデシリアル化コストをもたらします。

- エージェンティックコールにシーケンシャルなアプローチを使用すると、許容できないパフォーマンスへの影響があります。
- すべてが変更されたペイロードを返す同じペイロードで複数のエージェンティックコールを実行すると、各個別のエージェントが行ったペイロードへの変更を理解することが非常に困難で高価になります。
- 完全な OpenRTB ペイロードは、エージェントの目的に関係のない情報や、競合する情報や制限された情報を含む可能性のあるすべての情報を公開します。
- データプライバシーの目的でデータをスクラブする必要がある場合もあります。ペイロードを再構築するのは高価になります。
- エージェントが必要とする一部の情報は、OpenRTB 標準の一部ではない場合があります。
- オーケストレーションシステムにとって、なぜ 一部の変更が行われたのかを理解することは非常に難しい場合があります。

これらの変更に対処するために、この標準は次の要件を満たすプロトコルを想定しています。

1. 各エージェントは、最小データ原則に従って、オーケストレーターによって許可され、エージェントプロバイダーによって要求された最小限のデータのみを取得するべきです。特に、プライベートおよび競合シグナルは、エージェンティックな関与の前に削除されなければなりません。
2. OpenRTB ペイロードに対するエージェンティックな変更は、各変更を独立して評価または拒否できるように分離されなければなりません。
3. 各エージェントは、処理パイプラインの実装から切り離されるべきです。エージェントの実装は、呼び出しサイトやオーケストレーションシステムの内部構造などの内部の詳細を知らないべきです。
4. OpenRTB ペイロードに対するエージェンティックな変更は、オーケストレーターが提案された変更が行われた理由を理解できるように、その意図（intent）を宣言しなければなりません。

これらの要件を満たすために、この標準は、OpenRTB ビッドリクエストまたはビッドレスポンスへの変更を要求するための標準化されたプロトコルを提供するパッチメカニズムである OpenRTB パッチプロトコル を導入することによってモデルを拡張します。ビッドリクエストおよびビッドレスポンスを処理する場合、オーケストレーターはその処理に含めるべきコンテナを識別し、それぞれにリクエストを発行します。参加者のレスポンスは、希望するミューテーションを含むパッチオブジェクトを介してインフライトの変更を要求する場合があり、オーケストレーターは裁量でそれを受け入れるか拒否することができます。OpenRTB パッチプロトコルは、特定のミューテーションの目的を示す

宣言的な属性である intent（意図） の概念も導入しており、これによりオーケストレーターは、いつどのようにコンテナを呼び出すべきかをより適切に判断できます。

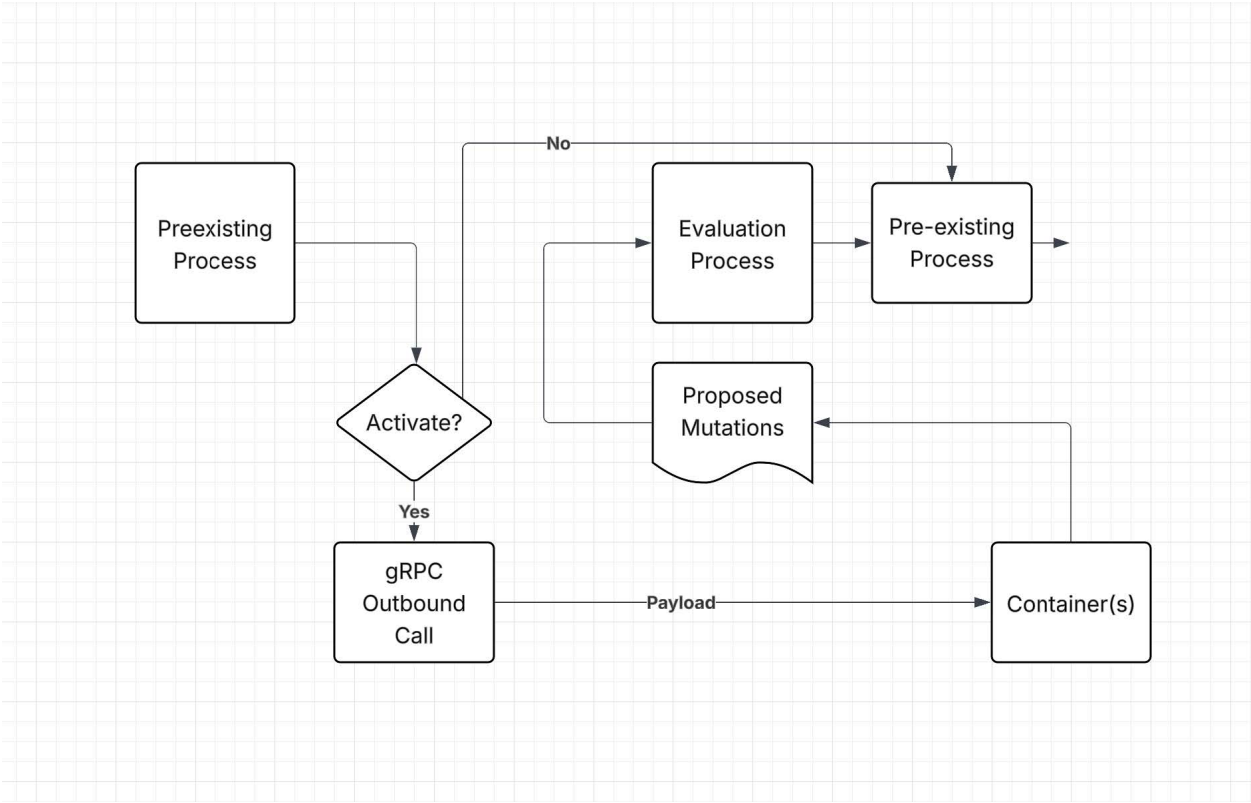

Figure1

パスの明確化

エージェントは、後の時点で挿入する必要があるミューテーションを返す場合があり、特定の ID（たとえば、特定の deal-id）を指定する場合があります。場合によっては、ミューテーションは、エージェントが予期する場所とは異なる場所または順序で挿入されることがあります。意図ごとのこの柔軟性を維持するために、ミューテーションパスは OpenRTB の概念から派生した意味的な参照を使用する場合があります。全体的な RTB 構造への明示的なパス（エージェントが知らない可能性があります）をたどるのではなく、パスは特定の JSON の場所ではなくビジネスレベルのエンティティを識別します。これにより、仕様はデータレイアウトではなくオークションの意味論の観点からミューテーションを表現できます。

なぜ gRPC と MCP なのか？

エージェントのこの仕様では、コンテナがオーケストレーターとエージェント間の通信にリモートプロシージャコール（RPC）をサポートする必要があります。このドキュメントのすべてのインターフ

ェースで protobuf を使用した gRPC が義務付けられていますが、適切な場合は MCP を使用できます。OpenRTB は最近、ネイティブレベルで gRPC サポートを追加しましたが、ほとんどの OpenRTB ペイロードは依然として REST および JSON です。REST ではなく gRPC をサポートする理由は次のとおりです。

1. これらのインターフェースでは、パフォーマンスが可能な限り高速である必要があります。gRPC は protobuf シリアル化を使用しており、JSON よりも容量と時間の効率がかなり高くなっています。
2. gRPC は検証が容易で、無効なペイロードの拒否機能が組み込まれています。
3. リモートプロシージャコールは、状態遷移よりもエージェントベースの通信に適したモデルです。

MCP も使用できます（そして将来的にはおそらく必要になるでしょう）。特に、MCP バージョン 2025-06-18 は、重要なトラフィックを処理するのに十分なパフォーマンスと安全性の両方を備えた最初のバージョンです。これは主に、ストリーム可能な HTTP と OAuth 認証のサポートによるものです。MCP は JSON-RPC も使用しているため、REST ベースのインターフェースの自然な後継となります。さらに、gRPC はサービス対サービスのエージェンティックオーケストレーションに理想的ですが、MCP はサービス対サービスオーケストレーションだけでなく、モデル対エージェントオーケストレーションにも使用でき、自律的なエージェンティックフローも可能にします。

どちらの場合も、同じインターフェースのセットが公開されます。gRPC では protobuf の rpc 定義を介して、MCP ではツール定義を介して公開されます。

Agent Manifest (エージェントマニフェスト)

エージェントマニフェストは、コンテナがオーケストレーターによってどのように使用および管理されるかを定義するラベルとして表現されたイメージメタデータを介して、コンテナイメージに追加される JSON の標準フィールドです。マニフェストは、キーが “agent-manifest” で、値が JSON 構造であるキーと値のペアとして指定されます。

マニフェストは、次のようなコンテナのビジネス要件を提供します。

- エージェントの名前
- エージェントのベンダー
- エージェントの所有者（電子メールアドレス）

また、いくつかのビジネスメタデータフィールドも含まれています。

- このエージェントによってサポートされる意図（intent）。

また、コンテナのランタイム構成に関する情報も含まれています。

- 必要な最小 CPU およびメモリリソース
- コンテナが通信する必要がある他のサービスへの依存関係（名前による）。

Technical Implementation (技術的実装)

Infrastructure Considerations (インフラストラクチャの考慮事項)

構成可能かつデプロイ可能であるということは、エージェントが基本インフラストラクチャと一貫した方法でさまざまなインフラストラクチャにデプロイできる必要があることを意味します。実際には、これは、それらが OCI コンテナとして構造化され、分散コンテナエコシステム（通常は Kubernetes）にデプロイ可能でなければならないことを意味します。

ビッドストリームデータに対して動作するエージェントについて、エージェントプロバイダーとオーケストレーターの両方が、Docker イメージに関する次のベストプラクティスに従わなければなりません。

- 堅牢な自己修復機能を確保するために、**必須のヘルスチェック**を liveness（生存）プローブと readiness（準備完了）プローブの両方として実装するべきです。これにより、インフラストラクチャが自律的に動作し、部分的な障害が発生した場合でも継続的なサービス可用性を維持できます。
- 開発から本番までの信頼の連鎖を確立し、許可されていないアーティファクトや侵害されたアーティファクトがビッドストリームエコシステムに侵入するのを防ぐために、すべてのデプロイメントで**イメージの署名と検証**を実施しなければなりません。
- **イメージは、管理され監査可能な CI/CD パイプラインによって構築されなければなりません。**パイプラインは、品質と一貫性を確保しながら速度を維持するために、統合されたセキュリティスキャン、コンプライアンスチェック、およびデプロイメント検証を使用して、リリースプロセス全体を自動化するべきです。
- サービスの提供に影響を与える前にパフォーマンスのボトルネックやセキュリティの異常をプロアクティブに特定できるように、**標準のログ記録と監視**をすべてのデプロイメントに統合して、システムヘルスのリアルタイムの可視性を提供しなければなりません。コンテナは、メト

リクスのための Open Telemetry エンドポイントをサポートし、分散トレースとスパン収集のための Open Tracing をサポートしなければなりません。

以下は、すべてのオーケストレーターが従わなければならない Docker イメージのベストプラクティスの一部です。

- 不正なワークロードから保護するために、サービス間、コンテナ間、およびプライバシーに配慮したサービス間のトラフィックフローを制御するための**ネットワークポリシーとセグメンテーション**を実装しなければなりません。横方向の移動を防ぎ、インフラストラクチャ内でのセキュリティ侵害の影響範囲を制限する多層防御の分離を提供するようにシステムを構成するべきです。
- 予測可能なパフォーマンスを保証し、リソースの枯渇を防ぎ、インフラストラクチャ全体の利用を最適化するために、名前空間レベルとコンテナレベルの両方で**リソースのクォータと制限**を定義しなければなりません。

Manifest Requirements (マニフェスト要件)

エージェントは、ニーズを表現する上記のようなマニフェストを提供することが期待されています。

1. 最小 CPU/メモリ要件
2. 呼び出す意図（intent）
3. 利用可能であることを期待するその他のサービス（名前による）

オーケストレーションエンティティがコンテナの要件と意図をサポートできない場合は、コンテナを開始しないべきです。

Bidstream Integration (ビッドストリーム統合)

エージェントは、サービスプロバイダーがコアビジネスロジックを実行できるようにする本質的にブラックボックスであるコンテナとして実装されます。各コンテナは、デプロイされると、切り離された分離されたコンピューティング環境に「住み」ます。これらのコンテナが機能するには、事前定義された意図を使用してビッドストリームに統合する必要があります。AdTech エコシステムの参加者は一般に、インフラストラクチャ、ソフトウェア、および慣行の独自の組み合わせを持っているためです。どの拡張ポイントがサポートされ、それらがエージェントとどのように統合されるかは、参加者ごとにカスタムになる可能性があります。全員が同じソフトウェアを実行することは実行可能でも望ましくもないため、代わりに、拡張ポイントとの一般的な相互作用をサポートする標準プロトコル

が定義されています。OpenRTB は、ビッドストリーム内のデータ通信に対する堅牢なサポートを提供し、ベンダー固有の拡張のための適切な抽象化を備えていますが、意思決定サイクルの一部としてコンテナがビッドストリームに伝播させたいような変更を組み込むための標準化されたサポートが不足しています。その不足に対処するために、この標準では、OpenRTB リクエストおよびレスポンスへの希望する変更を表現するためのプロトコルである「OpenRTB パッチ」を導入しています。

OpenRTB パッチは、次の原則に従うように設計されています。

- コンテナは、提供されたリクエストまたはレスポンスからの差分を記述する「パッチ」として、希望するミューテーションをオーケストレーターに提供します。これには、リクエストまたはレスポンスの特定の部分への追加、変更、および削除の組み合わせが含まれる場合があります。パッチが適用されるかどうかはオーケストレーターの裁量に委ねられており、オーケストレーターはビジネス要件に従ってミューテーションを受け入れるか拒否するかを決定します。オーケストレーターがパッチを受け入れるか拒否するかの選択をエージェントプロバイダーに通知するかどうか、およびその方法は、当事者に委ねられています。
- 各パッチはアトミックです。ミューテーションは全体として受け入れられるか、全体として拒否されなければなりません。複数のパッチは個別に受け入れまたは拒否される場合があります。ミューテーション間のトランザクションや順序保証はありません。
- ミューテーションは意味的に意味があります。OpenRTB パッチは、どのような変更が望ましいかだけでなく、なぜそれが要求されるかも指定します。これが意図（intent）として指定されます。
- OpenRTB パッチは、必要なプライバシーとシグナル情報をコンテナに伝播します。RTB リクエストで指定されたすべての動作を尊重することは、コンテナの責任です。
- OpenRTB パッチは、OpenRTB 標準のセマンティクスに従います。これには、オーケストレーターがエージェントプロバイダーに利用可能にしたいと考える非標準の拡張シグナリングのための「ext」オブジェクトの使用が含まれます。
- OpenRTB パッチは、要求されたパッチと適用されたパッチをログに記録することにより、オーケストレーターの環境内での監査可能性を促進します。
- オーケストレーターは、ミューテーションの受け入れ率と拒否理由を判断するための標準的なレポートとメトリクスを提供するべきです。

Service Naming (サービス命名)

前述のように、コンテナは依存しているサービスの名前に特定の依存関係を提供できます。これは多くの場合、オーケストレーターが提供する独自のサービス、または他のコンテナの名前にマップされるため、仕様は単に依存関係を示す文字列であることをマッピングのために指定するだけです。

API Design (API 設計)

gRPC と MCP の両方の RPC 定義には、従来の REST/JSON サービスよりも少し多くの構造が必要です。このドキュメントの目的のために、いくつかの主要なパターンが定義されていますが、これに関する正式な定義は、IAB Tech Lab の Github プロジェクトにある gRPC 定義です。これらの定義は、Protobuf 仕様の形式をとります。特に、サービスとメッセージは .proto ファイルで明確に表現されなければなりません。MCP ユースケースの場合、各拡張ポイントの RPC エンドポイントではなくツール定義であることを除けば、仕様は gRPC と同じです。

コメント期間の目的のために、初期の protobuf 定義があります。これらは変更される可能性があります。

Example Extension Point (拡張ポイントの例)

```
syntax = "proto2";

package com.iabtechlab.bidstream.mutation.v1; import "com/iabtechlab/openrtb/v2.6/openrtb.proto";

// service definition service
RTBExtensionPoint {
  // GetMutations returns RTBResponse containing mutations to be applied at the predetermined auction lifecycle event
  rpc GetMutations (RTBRequest) returns (RTBResponse);
}

message RTBRequest {
  // ENUM as per Programmatic Auction Definition IAB TL doc/spec
  required Lifecycle lifecycle = 1;
```

```
  required string id = 2;
  optional Extensions ext = 3;

  required com.iabtechlab.openrtb.v2.BidRequest bid_request = 4;
  optional com.iabtechlab.openrtb.v2.BidResponse bid_response = 5;

  required int32 tmax = 6;
}
```

これは拡張システムのバックボーンを形成します。個々のリクエストは、特定の拡張ポイントに対して単一の rpc と単一のストリームメカニズムを公開します。現在、仕様は、同じコンテナが複数の異なるサービスリクエストをサポートできるように、複数の呼び出しエンドポイントをサポートする必要性を予期しています。

AgenticRTB 固有の要件

Field	Type	Description
id	string	拡張ポイントリクエスト ID
bidRequest.imp	imp message	インプレッションメッセージ（OpenRTB に準拠、例外あり）
bidRequest.site	site message	サイトメッセージ（OpenRTB に準拠）
bidRequest.app	app message	アプリメッセージ（OpenRTB に準拠）
bidRequest.device	device message	デバイスメッセージ（OpenRTB に準拠）
bidRequest.user	user message	ユーザーメッセージ（OpenRTB に準拠）
bidRequest.regs	reg message	規制メッセージ（OpenRTB に準拠）
bidRequest.source	source message	ソースメッセージ（OpenRTB に準拠）
bidRequest.tmax	integer	タイムアウトを回避するため

		に、エクスチェンジがミューテーションの受信を許可する最大時間（ミリ秒単位）。インターネットのレイテンシを含む。

拡張レスポンス

ミューテーションは、既存のリクエストへの変更を表します。レコードを追加、削除、または更新することでシステムの状態を変更します。拡張プロバイダーのレスポンスは、これらのミューテーションの形式をとり、ビッドリクエストまたはレスポンスへの調整を提案します。拡張レスポンス gRPC の例を以下に示します。

```
message RTBResponse { // Or RequestPatch
  string id = 1;
  repeated mutation mutations = 2;
  MetaData metadata = 3;
}

message Mutation {
  string intent = 1;
  string op = 2;
  string path = 3;

  // The structure of value depends on the specified intent.
  // Reserve 100+ for intent-specific payloads
  // The structure of value depends on the specified intent.
  // Reserve 100+ for intent-specific payloads
  oneof value {
    // List of string Identifiers
    IDsPayload ids = 100;

    // Adjust properties of a specific deal
    AdjustDealPayload adjust_deal = 101;
```

```
      // Adjust the bid price
      AdjustBidPayload adjust_bid = 102;

      // Metrics or telemetry data
      AddMetricsPayload add_metrics = 103;
    }
}

message MetaData {
  string api_version = 1;
  string model_version = 2;
}
```

Example - Audience Segments - Activating Cohorts (例 - オーディエンスセグメント - コホートのアクティブ化)

以下は、レスポンスの戻り値の例です。gRPC はバイナリプロトコルであるため、シリアル化形式ではなく、概念を表現するために JSON 形式が使用されます。

```
{
  "intent": "activateSegments",
  "op": "add",
  "path": "/user/data/segment",
  "value": {
    "IDsPayload": ["18-35-age-segment","soccer-watchers"]
  }
}
```

Example - Complex Orchestration (例 - 複雑なオーケストレーション)

エンドポイントは複数のミューテーションを返す場合があります。その場合、各ミューテーションは個別に評価されます。複数のミューテーションにわたるトランザクションのサポートはないことに注意してください。ここの順序のミューテーションは、他のミューテーションとは独立して受け入れられるか拒否される場合があります。

```
[
  {
    "intent": "expireDeals",
    "op": "remove",
    "path": "/imp/1",
    "value": {
      "IDsPayload": ["deal100", "deal200"]
    }
  },
  {
    "intent": "activateDeals",
    "op": "add",
    "path": "/imp/1",
    "value": {
      "IDsPayload": ["deal300", "deal201"]
    }
  },
  {
    "intent": "adjustDeals",
    "op": "replace",
    "path": "/imp/2/deals/400",
    "value": {
      "AdjustDealPayload": {
        "bidfloor": 5.0,
        "wadomain": ["adomain.com"]
      }
    }
  },
  {
    "intent": "adjustDeals",
    "op": "replace",
    "path": "/imp/1/deals/500",
```

```
    "value": {
      "AdjustDealPayload": {
        "bidfloor": 3.5,
        "wadomain": ["example.com", "sample.com"]
      }
    }
  }
]
```

Manifest - Container.json (マニフェスト - Container.json)

これはコンテナ JSON の例です。これは、送信のために Docker イメージのメタデータに関連付けられていることに注意してください。

```
{
  "name": "openrtb-container-suite",
  "version": "1.0.0",
  "description": "Example container manifest for OpenRTB and Digital Advertising use cases
",
  "image": {
    "repository": "registry.example.com/rtb/auction-container",
    "tag": "v1.0.5",
    "digest": "sha256:abc123def4567890"
  },
  "resources": {
    "cpu": "500m",
    "memory": "256Mi"
  },
  "intents": [
    {
      "name": "bidResponseGeneration",
      "description": "Generate bid responses in real time based on request data"
    },
    {
```

```
      "name": "bidValuation",
      "description": "Evaluate incoming bid requests and determine bid amount"
    },
    {
      "name": "bidRequestModification",
      "description": "Propose mutations to a bid request prior to auction execution"
    },
    {
      "name": "auctionOrchestration",
      "description": "Route or prioritize bid requests across multiple buyers"
    },
    {
      "name": "metadataEnhancement",
      "description": "Insert or modify auction metadata such as fraud or viewability signals
"
    },
    {
      "name": "dynamicDealCuration",
      "description": "Curate deals in real time, optimize margins or enforce dynamic inclusi
on/exclusion lists"
    },
    {
      "name": "audienceSegmentation",
      "description": "Activate or enrich user cohorts and audience segments"
    }
  ],
  "dependencies": {
    "fraudDetectionService": {
      "service": "fraud-svc",
      "ENV_VARIABLE": FRAUD_URL
    },
    "audienceService": {
```

```
      "host": "audience-svc",
      "port": 9010
    }
  },
  "health": {
    "livenessProbe": {
      "httpGet": {
        "path": "/health/live",
        "port": 8080
      },
      "initialDelaySeconds": 10,
      "periodSeconds": 5
    },
    "readinessProbe": {
      "httpGet": {
        "path": "/health/ready",
        "port": 8080
      },
      "initialDelaySeconds": 5,
      "periodSeconds": 5
    }
  },
  "security": {
    "runAsNonRoot": true,
    "dropCapabilities": ["NET_ADMIN", "SYS_PTRACE"],
    "networkPolicies": {
      "ingress": ["fraud-svc", "audience-svc"],
      "egress": ["logging-svc"]
    }
  },
  "maintainers": [
    {
```

```
        "name": "IAB Tech Lab Example Team",
        "email": "support@iabtechlab.com"
      }
    ]
}
```