実証実験の進め方

実証実験の進め方

WordPress, CA Server 利用時は各社での実装は不要

- WordPress, CA Server 利用時
  - OP CIP 提供のツールだけで個別の開発は (原則) 不要
  - 各社で導入するプラグイン次第では改修が必要な場合もあるが通常不要
  - デプロイ先の PHP バージョンが古い場合は応相談
- その他の CMS 利用時 (あるいは WP でも CA Server を利用しない場合)
  - サイトの CMS を OP 対応する開発が必要
  - 手順についてはこの後で説明 (ドキュメントサイト参照)

WordPress サイトでの実証実験

WP サイトではプラグイン導入などだけで簡単に確認可能

- OP 登録をして OPS を受け取る
- CA Server を使って Site Profile を作成、設置
  - サイトの情報を JSON ファイルとして記載、curl 等で CA Server に送信
  - Site Profile を作成したら .well-known/sp.json に設置
- WP Plugin を導入してから記事・ページを更新
  - プラグイン導入後に更新・更改した記事・ページには自動的に CA Server を使って発行した CA が保存され、出力ページに埋め込まれます
- 拡張機能を入れたブラウザで動作を確認

Originator Profile 対応

Originator Profile 対応 (WP 以外)

既存 Web サイトを OP 利用サイトにするために必要なこと

- 対応/実装方針の検討 (事前検討)
  - 手動での実験的対応か、CMS を改修/実装するか、CA Server を利用するか...
- OP 登録 (事前手続き)
  - 組織情報や保有証明書を OP CIP に登録し、OP (OPS) の提供を受ける
- サイトプロファイルの設置 (サイト毎に一度)
  - サイト(ドメイン)毎のサイトプロファイル (SP) を作成し .well-known 配下に設置
- コンテンツ証明書の発行 (記事/コンテンツ毎に一度)
  - 記事/コンテンツに対応するコンテンツ証明書 (CA) を作成し CAS, OPS にリンク

記事作成/更新時の処理は WordPress を例とした参照実装を CIP 側で用意/更新

OP 対応: 対応/実装方針の検討

仕組みとイメージの確認だけなら手動対応での試験も可能

A. 手動での実験的対応をする場合 (最初の一歩として)

- 対応結果や仕組みの確認までであれば、CMS のプログラム改修は不要
- 各種証明書となる JSON ファイルの作成/設置とその参照用タグの挿入を手動で実施
- 各種証明書の JSON ファイルは CA Server (または CLI) で作成できます (手順は後述)

B. CMS として対応する場合 (推奨)

- CA Server を利用する場合 (推奨)
  - 鍵管理と署名を含む CA 生成処理をすべて CA Server に任せる事ができる
  - WordPress 利用サイトの場合は WP Plugin と CA Server の参照実装あり
- すべて自前で実装する場合 (個別相談にて対応)
  - 参照実装(Node パッケージや CLI 等)を利用または参考に CMS 側ですべて実装

OP 対応: OP 登録

情報発信者としての組織情報を登録し OP の発行を受ける

- OP 登録手続き (事前手続き)
  1. OP 登録サイト (+ Slack/Github など) を通じ組織情報を OP CIP に登録
  2. OP CIP が記載内容を確認し OP ID と Originator Profile (OP) を発行
  3. 合わせて PA Issuer の OP なども含む OPS (OP Set) も提供
- 補足: 鍵ペアの作成と CA Server の利用について
  - OP 登録時の鍵ペア作成と検証鍵(公開鍵)提出は CA Server 利用時は不要
  - CA Server 側で鍵ペア(署名鍵/検証鍵)の生成/管理を行う仕組みです

OP 対応: サイトプロファイルの設置

サイトの情報に署名したサイトプロファイルをサイト毎に作成・設置

- 詳細手順はドキュメント参照: https://docs.originator-profile.org/studies/general-instruction/sp-setup-guide/

- サイトプロファイルの作成と設置
  1. サイトの情報を記載した JSON ファイルを用意
  2. CA Server API CLI (あるいは CLI など) を使って署名した WSP を作成
  3. WSP と OPS をまとめたサイトプロファイル (JSON ファイル)を作成
  4. サイトプロファイルを /.well-known/sp.json に設置

OP 対応: コンテンツ証明書の発行

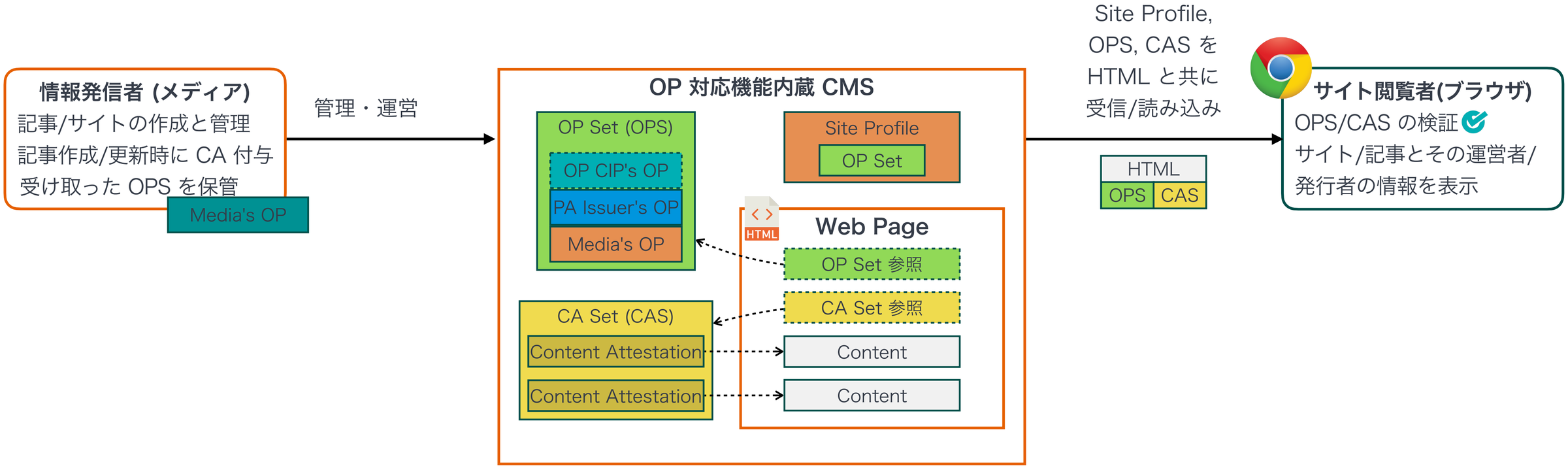
記事やコンテンツとその情報に署名した証明書を記事毎に作成・設置

- 詳細手順はドキュメント参照: https://docs.originator-profile.org/studies/general-instruction/cas-setup-guide/

- 記事/コンテンツの証明書 (CA: Content Attestation) の作成と設置
  1. CA を発行する対象記事 (を特定する CSS Selector) を決める
  2. コンテンツの情報を記載した JSON ファイルを用意
  3. CA Server API (あるいは CLI など) を使って署名した CA を作成
  4. ページ中の CA をすべてまとめた CAS (CA Set, JSON ファイル) を作成
  5. ページ HTML 中に CAS (と OPS) を参照する script タグを挿入

補: CAS と OPS は CA Server の API url を直接埋め込みも可能(非推奨)

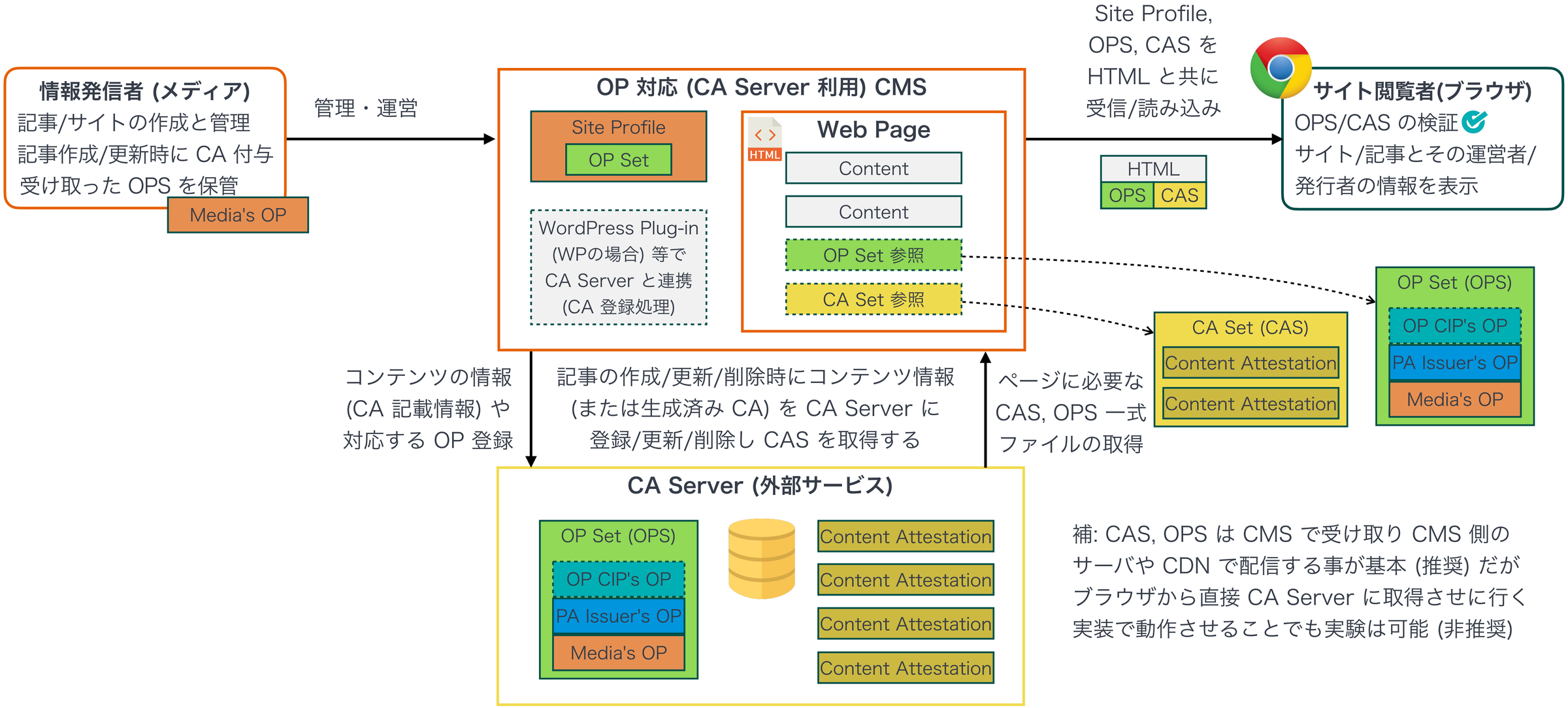
Originator Profile の対応 (CA Server なし)



記事の作成/更新時に CMS の機能として
(OGP や RSS 等の自動生成と同様に)
各コンテンツの CA を生成し CMS 内で保管

CA 発行者の OP を SP 内の OP Set に
含めているページでは個別に OP Set の
設置と参照を追加する必要はありません

Originator Profile の対応 (CA Server 利用)



補: CAS, OPS は CMS で受け取り CMS 側のサーバや CDN で配信する事が基本 (推奨) だがブラウザから直接 CA Server に取得させに行く実装で動作させることでも実験は可能 (非推奨)

CA Server の役割

CA の署名処理、署名鍵管理、CAS+OPS 作成

- OP 所有者の署名鍵を預かり CA の署名処理を担当
  - CA 記載情報を受け取り CA (W3C VC 2.0 形式) を作成
  - 署名鍵を安全に保管し CA への署名処理を行う
  - CA に対応する OP の登録を受け付け (OPS 作成用に) 保管
- ページに必要な CAS, OPS 一式ファイルを作成
  - ページに登録された CA と対応する OP のリストを管理
  - ページに登録された CA と OP をまとめた CAS, OPS ファイルを作成
  - (CAS, OPS ファイルのリクエストをブラウザから受けて送信)

記事作成/更新時の処理は WordPress を例とした参照実装を CIP 側で用意/更新

OP 対応関連ドキュメント

CMS の OP 対応を設計・検討頂く際の参考資料

- OP 対応実験一般の案内
  - https://docs.originator-profile.org/studies/general-instruction/
- サイト (サイト情報) の OP 対応 (Site Profile の作成と設置)
  - https://docs.originator-profile.org/studies/general-instruction/sp-setup-guide/
- 各コンテンツの OP 対応 (CA の作成と設置)
  - https://docs.originator-profile.org/studies/general-instruction/cas-setup-guide/
- OP 発行操作などを行う CLI ツールの README
  - https://github.com/originator-profile/profile-share/tree/main/apps/registry