Coding Prompt Cheat Sheet

For when you need to tell cursor or replit what a vCon is in a prompt...

Document for Coding Assistants: Structure of a vCon (Virtual Conversation)

Introduction

A **vCon** (Virtual Conversation) is a standardized JSON-based container designed to store and exchange data about real-time human conversations. It can encapsulate information from various communication modes, such as phone calls, video conferences, SMS, MMS, emails, and more. A vCon organizes this data into structured components for use in applications, data analysis, and regulatory compliance.

Key Components of a vCon

A vCon contains five main sections:

1. **Metadata**: Provides details about the conversation's context, including unique identifiers, timestamps, subject, and references to previous versions of the vCon.
2. **Parties**: Captures details about the participants in the conversation, including their roles, identifiers, and contact information.
3. **Dialog**: Stores the actual content of the conversation (e.g., text, audio, or video).
4. **Analysis**: Includes derived data such as transcripts, translations, sentiment analysis, or semantic tagging.
5. **Attachments**: Stores additional files related to the conversation, such as slides, images, or documents.

Structure of a vCon JSON Object

A vCon JSON object can be in one of three forms:

- **Unsigned**: Initial or intermediate state during data collection.
- **Signed**: Verified state with a digital signature for immutability.
- **Encrypted**: Secure state to protect sensitive data.

Below is a breakdown of the JSON keys and their values:

General Structure

```
{
  "vcon": "0.0.1",           // Syntax version
  "uuid": "string",         // Unique identifier
  "created_at": "date",     // Creation timestamp
  "updated_at": "date",     // Last modified timestamp
  "subject": "string",      // Optional topic of the conversation
  "redacted": {},           // Reference to the unredacted version (if a
  "appended": {},           // Reference to additional information (if a
  "group": [],              // Aggregation of multiple vCons
  "parties": [],            // Array of participant details
  "dialog": [],             // Array of conversation segments
  "analysis": [],           // Array of analysis data
  "attachments": []         // Array of attachment data
}
```

Metadata

- **vcon**: Syntax version of the vCon.
- **uuid**: A globally unique identifier for the vCon instance.
- **created_at**: Timestamp for when the vCon was created.
- **updated_at**: Timestamp for the last update to the vCon.
- **subject**: Free-text field describing the topic of the conversation.

Parties

Each participant in the conversation is represented as an object:

```
{
  "tel": "string",          // Telephone number
  "mailto": "string",       // Email address
  "name": "string",         // Participant's name
  "role": "string",         // Role in the conversation (e.g., agent, cus
  "uuid": "string"          // Unique identifier for the participant
}
```

Dialog

Each segment of the conversation is captured as a dialog object:

```
{
  "type": "string",         // Type (e.g., recording, text)
  "start": "date",          // Start timestamp
  "duration": "number",     // Duration in seconds
  "parties": [],             // Array of participant indices
  "mimetype": "string",     // MIME type of the content
  "filename": "string",     // Original filename (if applicable)
  "body": "string",         // Content (base64-encoded if inline)
  "encoding": "string"      // Encoding type (e.g., base64url)
}
```

Analysis

Derived insights about the conversation are stored in analysis objects:

```
{
  "type": "string",         // Analysis type (e.g., transcript, sentiment
  "dialog": [],              // Array of dialog indices related to the an
  "mimetype": "string",     // MIME type of the analysis file
  "filename": "string",     // Original filename
  "vendor": "string",       // Vendor or product name
  "body": "string",         // Content (base64-encoded if inline)
  "encoding": "string"      // Encoding type (e.g., base64url)
}
```

Attachments

Attachments provide supplemental data:

```
{
  "type": "string",         // Type or purpose of the attachment
  "start": "date",          // Timestamp when the attachment was exchange
  "party": "number",        // Index of the contributing participant
  "mimetype": "string",     // MIME type of the attachment
  "filename": "string",     // Original filename
  "body": "string",         // Content (base64-encoded if inline)
  "encoding": "string"      // Encoding type (e.g., base64url)
}
```

Security and Integrity

- **Signing**: vCons can be signed using JWS (JSON Web Signature) to ensure their integrity and authenticity.
- **Encryption**: Sensitive vCons can be encrypted using JWE (JSON Web Encryption) to protect their content.
- **Versioning**: Redacted and appended vCons reference their original versions to maintain a history of changes.

Example Use Cases

1. **Customer Support**: Storing call recordings, transcripts, and attachments for quality assurance and analytics.
2. **Legal Compliance**: Maintaining immutable records of conversations with signatures for regulatory purposes.
3. **Machine Learning**: Using vCon data as input for training AI models while adhering to data privacy laws.

Conclusion

This document outlines the structure and components of a vCon to guide your coding team in implementation. Follow the described JSON schema to ensure compliance with the vCon standard for storing and exchanging conversation data.