

リスト3.txt

```

001 async def stream(self, query_in: str,
002                     context_id: str) -> AsyncGenerator[dict[str, Any], None]:
003     # 解析フェーズ
004     yield {
005         "is_task_complete": False,
006         "require_user_input": False,
007         "content": "入力を解析しています...",
008     }
009
010     # 簡易的なコンテキストの維持
011     if context_id not in self.context_id2query:
012         self.context_id2query[context_id] = ""
013     self.context_id2query[context_id] += query_in + "\n"
014     query = self.context_id2query[context_id]
015
016     c_from, c_to, date_raw, parse_err = self._extract_params(query)
017     if parse_err:
018         yield {
019             "is_task_complete": False,
020             "require_user_input": True,
021             "content": f"入力解析に失敗しました: {parse_err}\n"
022                       "例: `USD EUR`、`"
023                       "`/` `usd jpy 2024-06-01`、`"
024                       "`/` `米ドルをユーロに 先週の金曜`、",
025         }
026         return
027
028     missing = []
029     if not c_from:
030         missing.append("通貨 (変換元, 3文字コード)")
031     if not c_to:
032         missing.append("通貨 (変換先, 3文字コード)")
033
034     if missing:
035         yield {
036             "is_task_complete": False,
037             "require_user_input": True,
038             "content": (
039                 "為替レートを取得するために次の情報が必要です: "
040                 f"[{', '.join(missing)}]。 \n"
041                 "例: `USD EUR` や `usd jpy 先週の金曜`",
042             ),
043         }
044         return
045
046     if c_from == c_to:
047         yield {
048             "is_task_complete": True,
049             "require_user_input": False,
050             "content": f"{c_from} と {c_to} は同一通貨です。為替レートは常に 1.0 です。",
051         }
052         return
053
054     # 日付正規化 (モデル解釈)
055     yield {
056         "is_task_complete": False,
057         "require_user_input": False,
058         "content": f"日付を解釈しています... (指定: {date_raw or 'latest'})",
059     }
060     date_norm, date_warn = self._normalize_date(date_raw or "latest")
061
062     # レート取得
063     yield {
064         "is_task_complete": False,
065         "require_user_input": False,
066         "content": f"為替レートを取得中... ({c_from} → {c_to}, 日付: {date_norm})",
067     }
068
069     data = get_exchange_rate(c_from, c_to, date_norm)
070
071     # 整形
072     yield {
073         "is_task_complete": False,
074         "require_user_input": False,
075         "content": "結果を整形しています...",
076     }
077
078     if "error" in data:
079         msg = f"取得に失敗しました: {data['error']}\n"
080         if date_warn:
081             msg += f"補足: {date_warn}\n"
082         msg += "通貨コード (例: USD EUR) と、必要なら自然言語で日付を指定して再実行してください。"
083         yield {
084             "is_task_complete": False,
085             "require_user_input": True,
086             "content": msg,
087         }
088         return

```

リスト3.txt

```

089
090 rate = data.get("rates", {}).get(c_to)
091 if rate is None:
092     yield {
093         "is_task_complete": False,
094         "require_user_input": True,
095         "content": "指定の通貨コードが不正か、レートが取得できませんでした。¥n"
096                     "例: `USD EUR 2024-06-01` や `USD EUR last Friday`。",
097     }
098     return
099
100 date_used = data.get("date", date_norm)
101 inv = 1.0 / rate if rate else None
102
103 lines = [
104     f"【為替レート】 {c_from} → {c_to}",
105     f"  日付: {date_used}",
106     f"- 1 {c_from} = {rate:.6f} {c_to}",
107 ]
108 if inv:
109     lines.append(f"- 1 {c_to} = {inv:.6f} {c_from}")
110 if date_warn:
111     lines.append(f"¥n注: {date_warn}")
112
113 yield {
114     "is_task_complete": True,
115     "require_user_input": False,
116     "content": "¥n".join(lines),
117 }

```