GENETRIX

THE CODE VAULT

20 Advanced SFMC Snippets

> *Most SFMC documentation covers the basics, but fails when you hit real-world enterprise architecture. These are the exact 20 snippets we use to bypass platform limitations, handle CRM sync quirks, and bulletproof high-volume operations.*

1. Parsing CRM Multi-Select Picklists

THE PROBLEM

Salesforce CRM syncs Multi-Select Picklists to SFMC as a single string separated by semicolons (e.g., "Camping;Hiking;Fishing"). You cannot use simple IF statements to check for individual preferences.

THE SOLUTION

We use `BuildRowsetFromString` to convert the delimited string into an iterable AMPScript rowset, allowing us to accurately query user preferences.

AMPSCRIPT

```
%%[
/* Incoming string from Sales Cloud */
VAR @hobbiesString, @hobbiesRowset, @rowCount, @i, @currentHobby
SET @hobbiesString = AttributeValue("CRM_Hobbies__c") /* e.g. "Camping;Hiking;Fishing" */

IF NOT Empty(@hobbiesString) THEN

    /* Convert the semicolon-separated string into a Rowset */
    SET @hobbiesRowset = BuildRowsetFromString(@hobbiesString, ";")
    SET @rowCount     = RowCount(@hobbiesRowset)

    FOR @i = 1 TO @rowCount DO
        SET @currentHobby = Field(Row(@hobbiesRowset, @i), 1)

        /* Output specific content based on the matched hobby */
        IF @currentHobby == "Hiking" THEN
            Output(Concat("<p>Check out our new trail boots!</p>"))
        ENDIF
    NEXT @i

ENDIF
]%%
```

2. Iterating JSON Arrays from API Events

THE PROBLEM

Journey Builder API events pass "Cart Items" as an array inside a JSON block. AMPScript cannot natively loop through JSON arrays. The JSON payload must first be set into an AMPScript variable before it can be accessed in SSJS.

THE SOLUTION

Set the JSON payload into a Variable using `Variable.SetValue` (or retrieve it from a DE field), then use an SSJS block to natively parse the array, iterate through the objects, and dynamically build standard HTML.

SERVER-SIDE JAVASCRIPT (SSJS)

```
// EntryJSON can be AttributeValue("DEfield") from a DE which stores JSON.
// Sample JSON payload:
// { "Cart": { "Items": [ {"Name":"Laptop","Price":75000}, {"Name":"Wireless Mouse","Price":1500} ] } }

<script runat="server">
    Platform.Load("core", "1");

    var EntryJSON = '{"Cart":{"Items":[{"Name":"Laptop","Price":75000},{"Name":"Wireless Mouse","Price":150
    Variable.SetValue("@JourneyEntryJSON", EntryJSON);
</script>

<script runat="server">
    Platform.Load("core", "1");

    try {
        var jsonPayload = Variable.GetValue("@JourneyEntryJSON");

        if (jsonPayload) {
            var parsedData = Platform.Function.ParseJSON(jsonPayload);
            var itemsArray = parsedData.Cart.Items; // Access the Items array
```

```
            if (itemsArray.length > 0) {
                for (var i = 0; i < itemsArray.length; i++) {
                    var productName  = itemsArray[i].Name;
                    var productPrice = itemsArray[i].Price;

                    // Write HTML directly to the email body
                    Write("<div><strong>" + productName + "</strong> - $" + productPrice + "</div>");
                }
            }
        }

    } catch (e) {
        Write(e.message);
    }
</script>
```

3. Dynamic Sender Profile Override

THE PROBLEM

You want emails to come "From" the customer's dedicated Sales Rep. Hardcoding this fails if the rep leaves or the customer has no assigned rep.

THE SOLUTION

Place this AMPScript block in a generic Content Block. Reference it in your Sender Profile setup to dynamically resolve the From Name and Email at send time, with a brand fallback.

AMPSCRIPT — SENDER PROFILE CONTENT BLOCK

```
%%[
/* Place this logic in a Content Block and call it via ContentBlockByKey() in the Sender Profile */

VAR @repName, @repEmail, @finalFromName, @finalFromEmail

/* Pull from Sendable DE or Contact Data */
SET @repName  = AttributeValue("AccountOwner_Name")  // e.g. "Akash"
SET @repEmail = AttributeValue("AccountOwner_Email") // e.g. "akash@yop.com"

/* Validate that BOTH fields exist and the email is formatted correctly */
IF NOT Empty(@repName) AND IndexOf(@repEmail, "@") > 0 THEN
    SET @finalFromName  = @repName
    SET @finalFromEmail = @repEmail
ELSE
    /* Fallback to default brand sender */
    SET @finalFromName  = "Genetrix Support"
    SET @finalFromEmail = "hello@genetrix.tech"
ENDIF
]%%
```

4. Advanced SQL Deduplication (Partitioning)

THE PROBLEM

A standard SQL `GROUP BY` removes duplicates but gives you no control over which record it keeps. You often lose the most recently updated data.

THE SOLUTION

Using `ROW_NUMBER() OVER (PARTITION BY...)` allows you to group records by Email Address, order them by Modified Date, and strictly select the absolute newest record.

SQL

```
SELECT
    x.SubscriberKey,
    x.EmailAddress,
    x.FirstName,
    x.LastModifiedDate
FROM
(
    SELECT
        SubscriberKey,
        EmailAddress,
        FirstName,
        LastModifiedDate,
        ROW_NUMBER() OVER (
            PARTITION BY EmailAddress       /* group by the duplicate field */
            ORDER BY     LastModifiedDate DESC /* newest record gets rank 1   */
        ) AS RowNum
    FROM Staging_Data_Extension
) x
WHERE RowNum = 1
```

5. Create a CRM Task via AMPScript

THE PROBLEM

You want a sales rep to call a customer immediately after the customer clicks a specific link in an email.

THE SOLUTION

Direct the email link to a CloudPage. On the CloudPage, run this AMPScript to use the SFMC Connector to natively create a Task in Sales Cloud assigned to the owner.

AMPSCRIPT — CLOUDPAGE

```
%%[
/* Grab the subscriber data passed via the email link */
VAR @contactId, @createTask
SET @contactId = RequestParameter("ContactKey")

/* Ensure we have an ID before executing */
IF NOT Empty(@contactId) THEN

    /* CreateSalesforceObject(ObjectType, FieldCount, Field1, Value1, ...) */
    SET @createTask = CreateSalesforceObject(
        "Task", 4,
        "WhoId",    @contactId,
        "Subject",  "High Intent Click: Follow Up Required",
        "Status",   "Open",
        "Priority", "High"
    )

ENDIF
]%%
```

6. Real-Time RSS/XML Feed Parsing

THE PROBLEM

You send a weekly newsletter and manually copy/paste blog post titles and links every week. It is tedious and prone to user error.

THE SOLUTION

Use `HTTPGet` and `BuildRowsetFromXML` to dynamically fetch your website's RSS feed at send time and inject the latest 3 articles automatically.

AMPSCRIPT

```
%%[
/* Fetch the live RSS feed */
VAR @xml, @nodes, @rowCount, @i, @title, @link
SET @xml      = HTTPGet("https://genetrix.tech/feed/")

/* Each <item> node in the RSS feed becomes one row in the rowset */
SET @nodes    = BuildRowsetFromXML(@xml, "//item", 1)
SET @rowCount = RowCount(@nodes)

/* Cap at the 3 most recent posts */
IF @rowCount > 3 THEN
    SET @rowCount = 3
ENDIF

IF @rowCount > 0 THEN
    FOR @i = 1 TO @rowCount DO

        /* Field names match the XML element names inside <item> */
        SET @title = Field(Row(@nodes, @i), "title")
        SET @link  = Field(Row(@nodes, @i), "link")
]%%

<h3><a href="%%=v(@link)=%%">%%=v(@title)=%%</a></h3>

%%[
    NEXT @i
ENDIF
]%%
```

7. Triggering a Journey via SSJS (API)

THE PROBLEM

You have a custom form on a CloudPage and need to inject the user into an API-Driven Journey Builder flow the second they hit submit.

THE SOLUTION

Use SSJS to construct a JSON payload and fire a POST request to the SFMC REST API interaction endpoint. Ensure `@SubscriberKey` and `@Email` are set as AMPScript variables before this block executes.

SERVER-SIDE JAVASCRIPT (SSJS) — CLOUDPAGE

```
<script runat="server">
    Platform.Load("core", "1");

    // authToken must be retrieved separately via the auth endpoint (see Snippet 11)
    var endpoint          = "https://YOUR_SUBDOMAIN.rest.marketingcloudapis.com/interaction/v1/events";
```

```
    var eventDefinitionKey = "APIEvent-XYZ123";
    var subscriberKey     = Variable.GetValue("@SubscriberKey");

    var payload = {
        ContactKey:         subscriberKey,
        EventDefinitionKey: eventDefinitionKey,
        Data: {
            EmailAddress: Variable.GetValue("@Email"),
            FormSource:   "CloudPage"
        }
    };

    var req = new Script.Util.HttpRequest(endpoint);
    req.emptyContentHandling = 0;
    req.retries              = 2;
    req.continueOnError      = true;
    req.contentType          = "application/json";
    req.setHeader("Authorization", "Bearer " + authToken);
    req.method               = "POST";
    req.postData             = Stringify(payload);

    var resp = req.send();
</script>
```

8. Phone Number Normalizing

THE PROBLEM

MobileConnect requires strict formatting (Country Code + Number). CRM data often comes over looking like `(555) 123-4567`, which will fail an SMS send.

THE SOLUTION

Use AMPScript's native `Replace` function to strip out all spaces, parentheses, and dashes, ensuring a clean string before appending the country code.

AMPSCRIPT

```
%%[
VAR @rawPhone, @cleanPhone, @finalPhone
SET @rawPhone = AttributeValue("Phone")

IF NOT Empty(@rawPhone) THEN

    /* Strip common non-digit characters one by one */
    SET @cleanPhone = Replace(@rawPhone,   " ", "")
    SET @cleanPhone = Replace(@cleanPhone, "(", "")
    SET @cleanPhone = Replace(@cleanPhone, ")", "")
    SET @cleanPhone = Replace(@cleanPhone, "-", "")
    SET @cleanPhone = Replace(@cleanPhone, "+", "")
    SET @cleanPhone = Replace(@cleanPhone, ".", "")

    /* Ensure it is 10 digits before appending the US country code */
    IF Length(@cleanPhone) == 10 THEN
        SET @finalPhone = Concat("1", @cleanPhone)
    ELSE
        SET @finalPhone = "" /* handle invalid number gracefully */
    ENDIF

ENDIF
]%%
```

9. SSJS "Safe Send" Error Logger

THE PROBLEM

A faulty AMPScript lookup can cause a multi-million record send job to crash and fail midway through.

THE SOLUTION

Wrap complex execution logic in an SSJS try/catch block. If an error occurs, it suppresses the crash, renders a fallback content block, and logs the specific system error to an Audit DE.

SERVER-SIDE JAVASCRIPT (SSJS)

```
<script runat="server">
    Platform.Load("core", "1");

    var subKey = Variable.GetValue("_subscriberkey");
    var jobID  = Variable.GetValue("jobid");

    try {
        // Risky execution — e.g. a complex external API call
        var externalData = Platform.Function.HTTPGet("https://api.example.com/data?id=" + subKey);
        Write(externalData.Content);

    } catch (err) {

        // 1. Prevent send failure — show fallback content block
```

```
        Write(Platform.Function.ContentBlockByKey("Generic_Fallback_Block"));

        // 2. Log the exact error to a DE so admins can investigate
        var errorDesc = String(err.description);
        Platform.Function.InsertDE(
            "Send_Error_Logs",
            ["SubscriberKey", "JobID", "ErrorMsg"],
            [subKey, jobID, errorDesc]
        );
    }
</script>
```

10. High-Volume Send Logging (InsertDE)

THE PROBLEM

Using the standard `InsertData()` AMPScript function during a send of 500,000+ emails causes database table locking and massive send delays.

THE SOLUTION

Use `InsertDE()` instead. It queues the insert asynchronously at the server level, preventing table locks and keeping send speeds incredibly fast.

AMPSCRIPT

```
%%[
/* Gather send-time context variables */
VAR @subKey, @jobID, @dynamicContentID
SET @subKey           = AttributeValue("_subscriberkey")
SET @jobID            = AttributeValue("jobid")
SET @dynamicContentID  = "Promo_A_Variant"

/* InsertDE is asynchronous — it does not lock the table during batch sends.
   Format: InsertDE("TargetDE", "Col1", Val1, "Col2", Val2, ...) */
InsertDE(
    "High_Volume_Send_Log",
    "SubscriberKey", @subKey,
    "JobID",         @jobID,
    "ContentServed", @dynamicContentID,
    "Timestamp",     Now()
)
]%%
```

11. Generate REST API Bearer Token

THE PROBLEM

You need to make a REST API call to SFMC from a CloudPage, but you cannot hardcode a static token because they expire every 20 minutes.

THE SOLUTION

Use SSJS to securely authenticate using your Server-to-Server Client ID and Secret to generate a fresh Bearer token on demand.

SERVER-SIDE JAVASCRIPT (SSJS) — CLOUDPAGE

```
<script runat="server">
    Platform.Load("core", "1");

    var tenantSubdomain = "YOUR_SUBDOMAIN";
    var clientId        = "YOUR_CLIENT_ID";
    var clientSecret    = "YOUR_CLIENT_SECRET";

    var authEndpoint = "https://" + tenantSubdomain + ".auth.marketingcloudapis.com/v2/token";

    var payload = {
        grant_type:    "client_credentials",
        client_id:     clientId,
        client_secret: clientSecret
    };

    var req = new Script.Util.HttpRequest(authEndpoint);
    req.emptyContentHandling = 0;
    req.retries              = 2;
    req.continueOnError      = true;
    req.contentType          = "application/json";
    req.method               = "POST";
    req.postData             = Stringify(payload);

    var resp          = req.send();
    var tokenResponse = Platform.Function.ParseJSON(String(resp.content));
    var bearerToken   = tokenResponse.access_token;
</script>
```

12. Bulk Data Upsert via WSProxy

THE PROBLEM

Standard SSJS `Rows.Add()` only supports inserts, not upserts, and is significantly slower for high-volume operations on a CloudPage.

THE SOLUTION

WSProxy bypasses the Core library overhead, allowing you to batch process and upsert thousands of records in fractions of a second.

SERVER-SIDE JAVASCRIPT — WSPROXY

```
<script runat="server">
    Platform.Load("core", "1");

    var prox = new Script.Util.WSProxy();

    // Build the array of records to upsert
    var records = [
        {
            CustomerKey: "Target_DE_External_Key",
            Properties: [
                { Name: "SubscriberKey", Value: "0031A00000XYZ" },
                { Name: "Status",        Value: "Active"        }
            ]
        },
        {
            CustomerKey: "Target_DE_External_Key",
            Properties: [
                { Name: "SubscriberKey", Value: "0031A00000ABC" },
                { Name: "Status",        Value: "Inactive"      }
            ]
        }
    ];

    // SaveAction "UpdateAdd" = upsert (update if exists, insert if not)
    var options = { SaveOptions: [{ PropertyName: "*", SaveAction: "UpdateAdd" }] };
    var res     = prox.updateBatch("DataExtensionObject", records, options);
</script>
```

13. Secure URL Parameter Passing

THE PROBLEM

Passing a SubscriberKey or Email Address in plain text via a URL parameter to a CloudPage violates PII security policies.

THE SOLUTION

Use the native AMPScript `CloudPagesURL` function. It automatically encrypts URL parameters, providing a secure and straightforward way to pass data to a CloudPage.

AMPSCRIPT — EMAIL & CLOUDPAGE

```
/* ─── IN THE EMAIL ──────────────────────────────────────────────── */
%%[
VAR @subKey
SET @subKey = AttributeValue("_subscriberkey")
]%%

<!-- Replace 123 with your actual CloudPage ID -->
<a href="%%=RedirectTo(CloudPagesURL(123, 'subKey', @subKey))=%%">Update Profile</a>


/* ─── ON THE CLOUDPAGE ──────────────────────────────────────────── */
%%[
VAR @incomingData
SET @incomingData = RequestParameter("subKey")
]%%
```

14. Triggering an Automation on Demand

THE PROBLEM

A user submits a form on a CloudPage, and you need to instantly refresh an SQL query or execute a data extract, rather than waiting for the next scheduled hourly run.

THE SOLUTION

Use WSProxy to locate the specific Automation by its External Key and force a "Perform" action to start it immediately.

SERVER-SIDE JAVASCRIPT — WSPROXY

```
<script runat="server">
    Platform.Load("core", "1");

    var prox                  = new Script.Util.WSProxy();
    var automationCustomerKey = "Data_Refresh_Auto_Key";

    try {
        var action = "Start";
        var props  = { CustomerKey: automationCustomerKey };
        var res    = prox.performItem("Automation", props, action);
```

```
        if (res.Status == "OK") {
            Write("Automation started successfully.");
        }

    } catch (e) {
        Write("Error: " + Stringify(e));
    }
</script>
```

15. Rendering Stored HTML via AMPScript

THE PROBLEM

You are pulling a raw HTML string (like a dynamic banner code) from a Data Extension, but SFMC prints the literal code tags onto the screen instead of rendering the image/banner.

THE SOLUTION

Use the `TreatAsContent` function. It forces the SFMC rendering engine to evaluate the string as active code rather than plain text.

AMPSCRIPT

```
%%[
/* Assume the DE field contains raw HTML, e.g. <img src="promo.jpg"> */

VAR @storedHTML, @renderedOutput
SET @storedHTML = Lookup("Content_Library_DE", "HTML_Code", "PromoID", "123")

IF NOT Empty(@storedHTML) THEN
    /* Forces the SFMC engine to render the stored HTML tags */
    SET @renderedOutput = TreatAsContent(@storedHTML)
ENDIF
]%%

/* Output the processed block */
%%=v(@renderedOutput)=%%
```

16. SQL: High-Frequency Incremental Pulls

THE PROBLEM

You have an automation running every 15 minutes to process new signups. Standard date checks often cause overlapping records or missed data if the job delays.

THE SOLUTION

Compare the record's creation date directly against the current system time minus exactly 15 minutes using `DATEADD`.

SQL

```
SELECT
    SubscriberKey,
    EmailAddress,
    CreatedDate
FROM
    Master_Subscribers_DE
WHERE
    /* Pull only records created within the last 15 minutes of execution */
    CreatedDate >= DATEADD(minute, -15, GETUTCDATE())
```

17. Generate Unique GUIDs for Webhooks

THE PROBLEM

You are pushing data from SFMC to an external third-party webhook. The receiving system requires a perfectly unique Transaction ID for every single push to prevent duplicates.

THE SOLUTION

Use the native AMPScript `GUID()` function to generate an instant, universally unique identifier string at send time.

AMPSCRIPT

```
%%[
/* Generates a standard 36-character UUID, e.g. 123e4567-e89b-12d3-a456-426614174000 */

VAR @transactionID
SET @transactionID = GUID()

/* Log the unique ID to a DE alongside the subscriber key and send date */
InsertDE(
    "Webhook_Logs",
    "SubscriberKey", AttributeValue("_subscriberkey"),
    "TransactionID", @transactionID,
    "SentDate",      Now()
)
]%%
```

18. Base64 Encoding for JSON Payloads

THE PROBLEM

Passing complex text (like rich text formatting, quotes, or deep URLs) inside a JSON payload often breaks the syntax formatting.

THE SOLUTION

Convert the problematic string to Base64 using SSJS before injecting it into the JSON. The receiving system can easily decode it back to a clean string.

SERVER-SIDE JAVASCRIPT (SSJS)

```
<script runat="server">
    Platform.Load("core", "1");

    var rawString = '<h1 style="color:red;">Warning!</h1>';

    // Encode using the native SSJS utility (no Platform.Function prefix required)
    var encodedString = Base64Encode(rawString);

    // Inject into the JSON payload — now structurally safe
    var payload = {
        messageType: "alert",
        contentData: encodedString
    };

    Write(Stringify(payload));
</script>
```

19. Calculating Dynamic Fiscal Quarters

THE PROBLEM

You need an email to dynamically state the current Fiscal Quarter (Q1, Q2, etc.) based on the date the email is sent, without manually updating content blocks.

THE SOLUTION

Extract the numeric month from the system date and use an AMPScript IF statement to dynamically assign the correct string.

AMPSCRIPT

```
%%[
/* FormatDate with "MM" returns the zero-padded numeric month (01-12) */
VAR @currentMonth, @currentQuarter
SET @currentMonth = FormatDate(Now(), "MM")

IF     @currentMonth == "01" OR @currentMonth == "02" OR @currentMonth == "03" THEN
    SET @currentQuarter = "Q1"
ELSEIF @currentMonth == "04" OR @currentMonth == "05" OR @currentMonth == "06" THEN
    SET @currentQuarter = "Q2"
ELSEIF @currentMonth == "07" OR @currentMonth == "08" OR @currentMonth == "09" THEN
    SET @currentQuarter = "Q3"
ELSE
    SET @currentQuarter = "Q4"
ENDIF
]%%

Here is your %%=v(@currentQuarter)=%% Performance Review.
```

20. Handling Null Values in SQL Safely

THE PROBLEM

A key field being NULL can produce unexpected output in your query results, such as missing or incorrect data for that subscriber.

THE SOLUTION

Use the `ISNULL()` or `COALESCE()` function to force a default fallback value during the query execution, preserving the dataset integrity.

SQL

```
SELECT
    A.SubscriberKey,
    A.EmailAddress,

    /* If FirstName is NULL, substitute the string 'Customer' as fallback */
    ISNULL(A.FirstName, 'Customer') AS FirstName,

    /* COALESCE returns the first non-null value from the list */
    COALESCE(B.MobileNumber, A.Phone, 'No Phone Provided') AS ContactNumber

FROM
    Master_Subscribers A
LEFT JOIN
    SMS_Opt_Ins B ON A.SubscriberKey = B.SubscriberKey
```

© 2026 Genetrix Technology. All rights reserved.

Deep Salesforce Expertise. No Fluff.

www.genetrix.tech