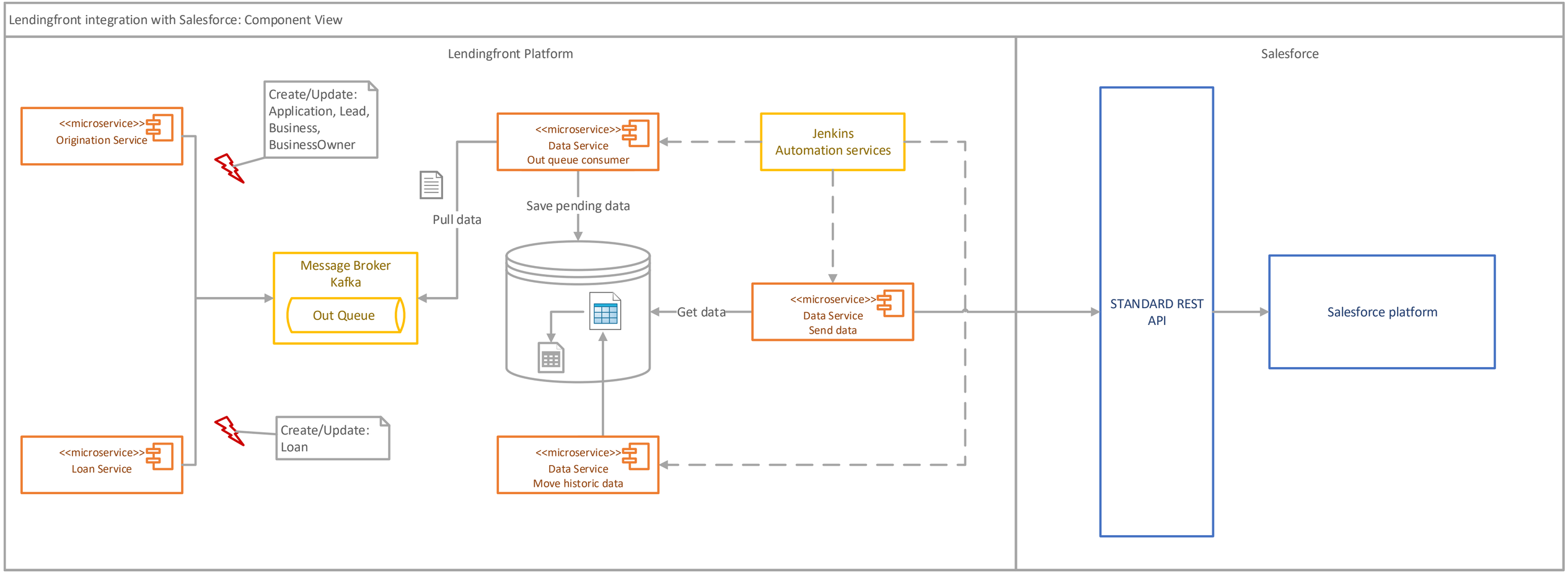
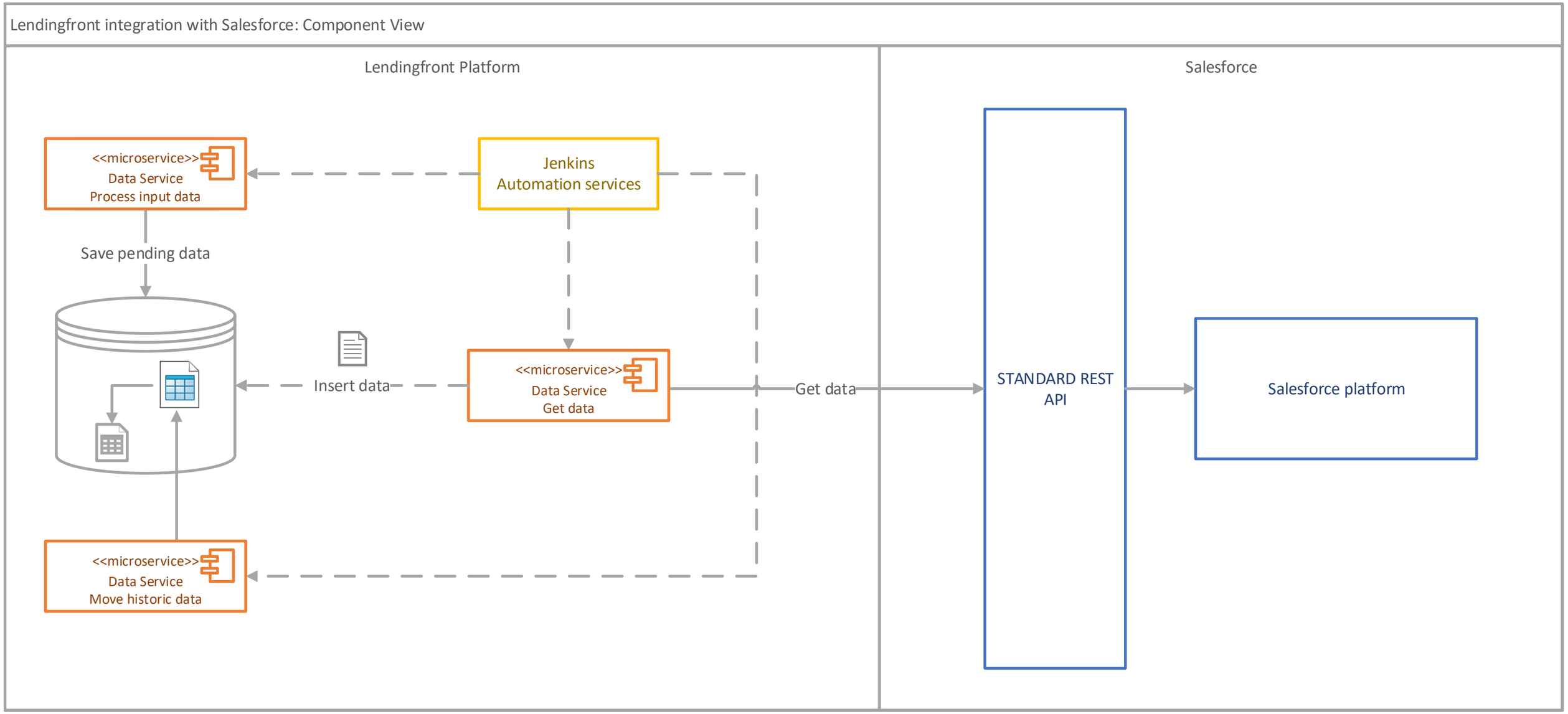
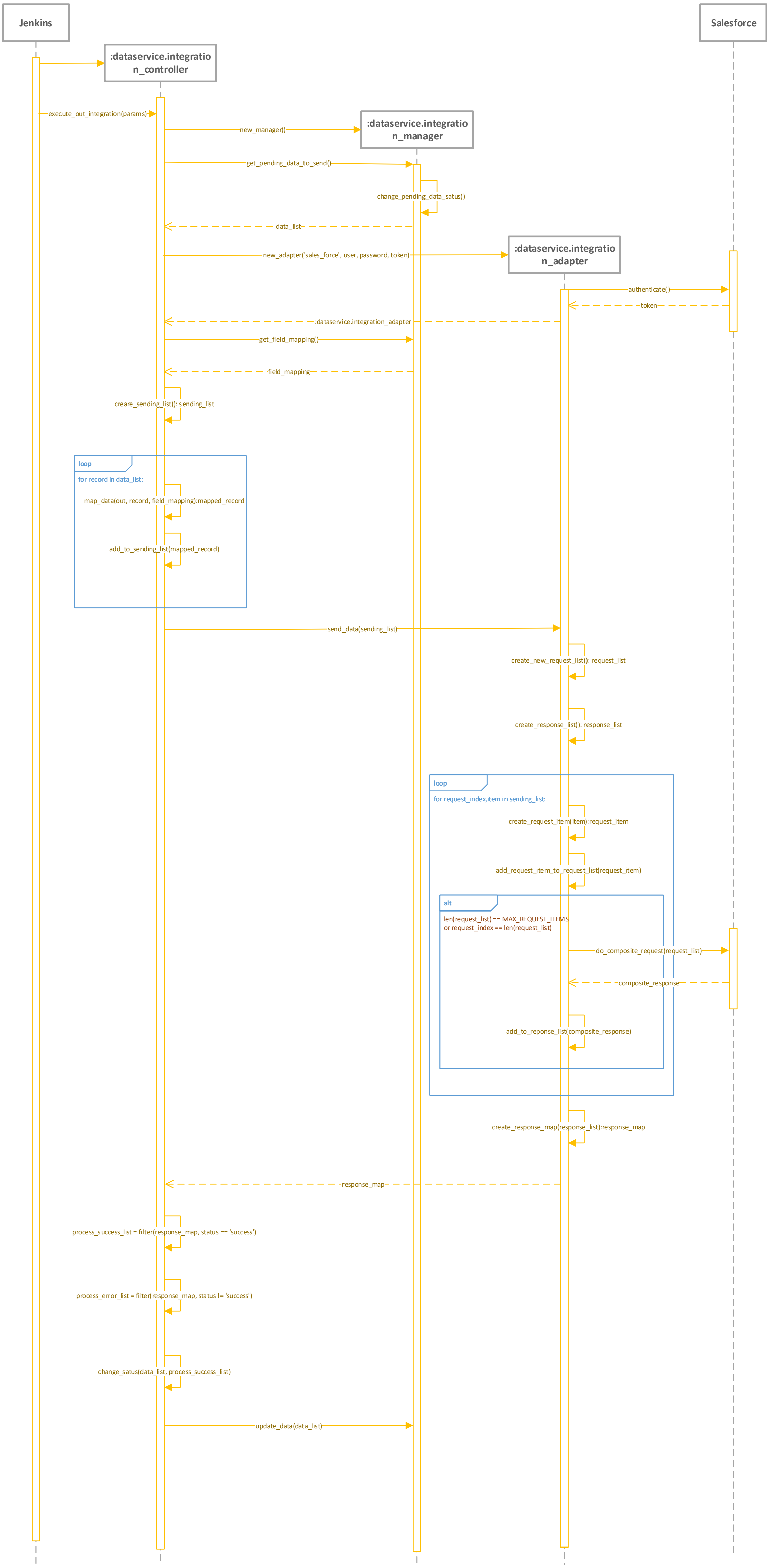
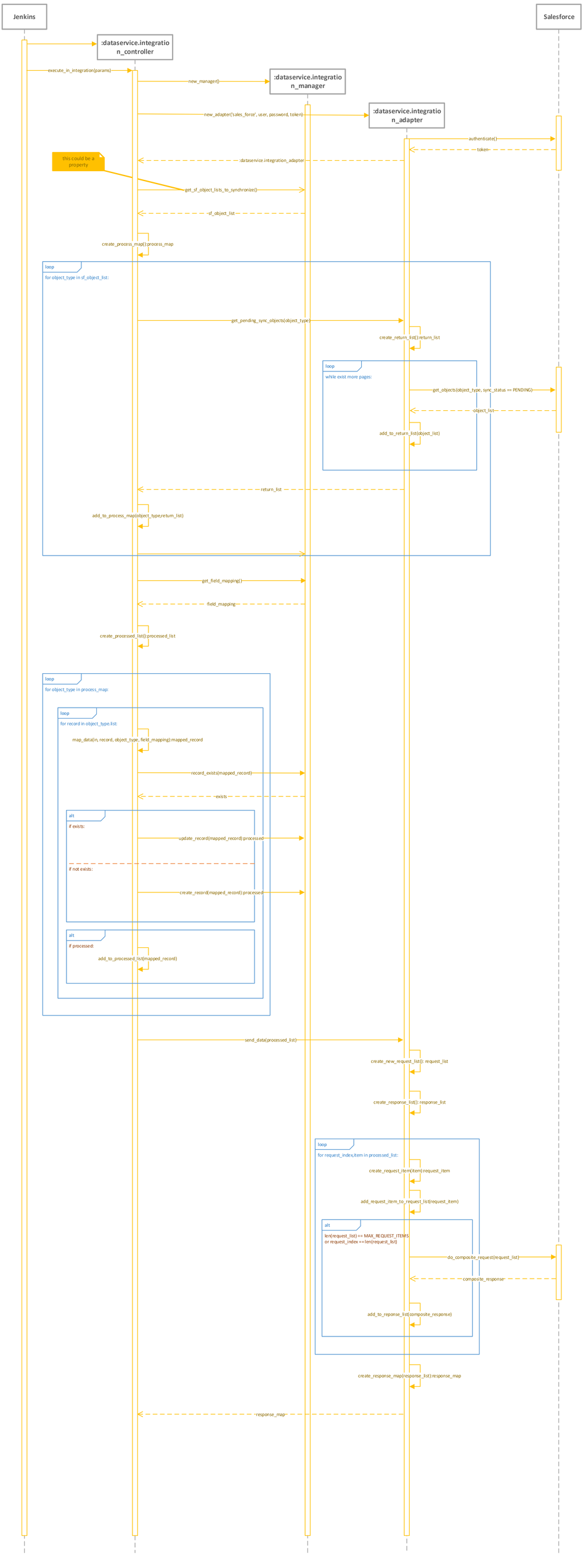

Lendingfront integration with Salesforce: Component View
Lendingfront Platform
Salesforce
<<microservice>>
Origination Service
Create/Update:
Application, Lead,
Business,
BusinessOwner
<<microservice>>
Data Service
Out queue consumer
Jenkins
Automation services
Pull data
Save pending data
Message Broker
Kafka
Out Queue
Get data
<<microservice>>
Data Service
Send data
STANDARD REST
API
Salesforce platform
<<microservice>>
Loan Service
Create/Update:
Loan
<<microservice>>
Data Service
Move historic data


Lendingfront integration with Salesforce: Component View
Lendingfront Platform
Salesforce
<<microservice>>
Data Service
Process input data
Jenkins
Automation services
Save pending data
Insert data
<<microservice>>
Data Service
Get data
Get data
STANDARD REST
API
Salesforce platform
<<microservice>>
Data Service
Move historic data


Jenkins
:dataservice.integration_controller
Salesforce
execute_out_integration(params)
new_manager()
:dataservice.integration_manager
get_pending_data_to_send()
change_pending_data_satus()
data_list
new_adapter('sales_force', user, password, token)
:dataservice.integration_adapter
authenticate()
token
:dataservice.integration_adapter
get_field_mapping()
field_mapping
creare_sending_list(): sending_list
loop
for record in data_list:
map_data(out, record, field_mapping):mapped_record
add_to_sending_list(mapped_record)
send_data(sending_list)
create_new_request_list(): request_list
create_response_list(): response_list
loop
for request_index,item in sending_list:
create_request_item(item):request_item
add_request_item_to_request_list(request_item)
alt
len(request_list) == MAX_REQUEST_ITEMS
or request_index == len(request_list)
do_composite_request(request_list)
composite_response
add_to_reponse_list(composite_response)
create_response_map(response_list):response_map
response_map
process_success_list = filter(response_map, status == 'success')
process_error_list = filter(response_map, status != 'success')
change_satus(data_list, process_success_list)
update_data(data_list)


Jenkins
Salesforce
:dataservice.integration_controller
:dataservice.integration_manager
:dataservice.integration_adapter
execute_in_integration(params)
new_manager()
new_adapter('sales_force', user, password, token)
authenticate()
token
this could be a property
:dataservice.integration_adapter
get_sf_object_lists_to_synchronize()
sf_object_list
create_process_map():process_map
loop
for object_type in sf_object_list:
get_pending_sync_objects(object_type)
create_return_list():return_list
loop
while exist more pages:
get_objects(object_type, sync_status == PENDING)
object_list
add_to_return_list(object_list)
return_list
add_to_process_map(object_type,return_list)
get_field_mapping()
field_mapping
create_processed_list():processed_list
loop
for object_type in process_map:
loop
for record in object_type.list:
map_data(in, record, object_type, field_mapping):mapped_record
record_exists(mapped_record)
exists
alt
if exists:
update_record(mapped_record):processed
if not exists:
create_record(mapped_record):processed
alt
if processed:
add_to_processed_list(mapped_record)
send_data(processed_list)
create_new_request_list():request_list
create_response_list():response_list
loop
for request_index,item in processed_list:
create_request_item(item):request_item
add_request_item_to_request_list(request_item)
alt
len(request_list) == MAX_REQUEST_ITEMS
or request_index == len(request_list)
do_composite_request(request_list)
composite_response
add_to_reponse_list(composite_response)
create_response_map(response_list):response_map
response_map